MASTER'S THESIS

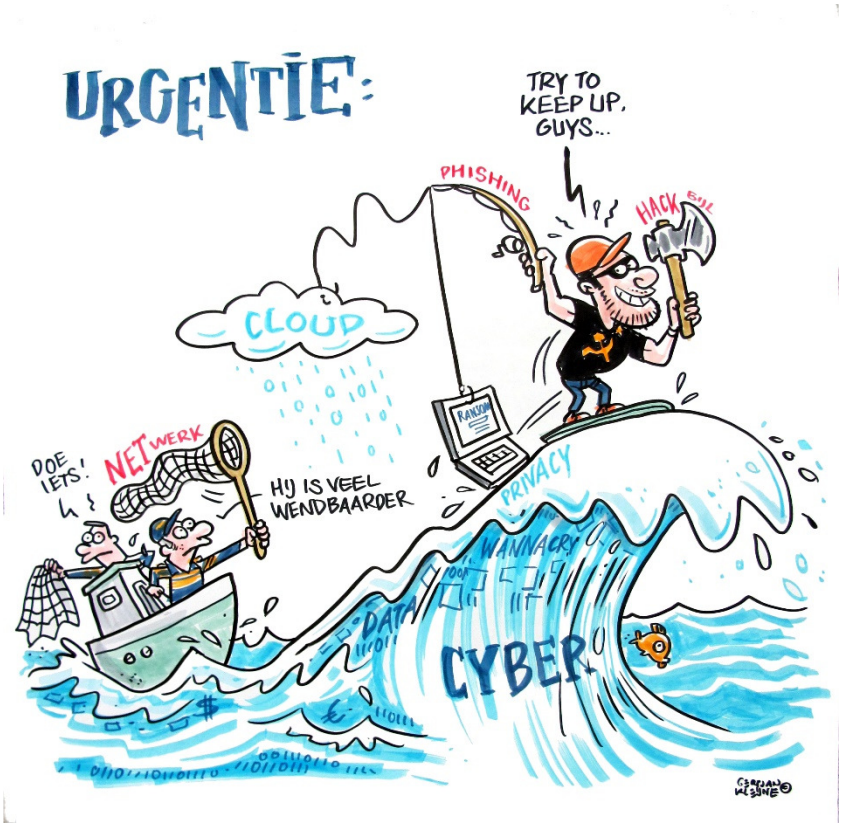
DNS: De Achilleshiel van flux botnets?

Detectie van flux botnets met recursive DNS data met behulp van Machine Learning

van der Meulen, E.M.

Award date:
2021

DNS: DE ACHILLESHIEL VAN FLUX BOTNETS?



29/6/2021

Detectie van flux botnets met recursive DNS data met behulp van Machine Learning

Else Maria van der Meulen,

DNS: De Achilleshiel van flux botnets?

DETECTIE VAN FLUX BOTNETS MET RECURSIVE DNS DATA MET BEHULP VAN MACHINE LEARNING

DNS: The Achilles Heel of flux botnets?
Detection of Flux Botnets with recursive DNS data using Machine Learning

Afstudeerscriptie van:
E.M. van der Meulen

Open University of the Netherlands, Faculty of Science
Master's Programme in Computer Science

Student	: Else Maria van der Meulen
Student nummer	
Cursuscode	: IM990C
1st Supervisor	: Dr. Ir. H.P.E. Vranken
2nd Supervisor	: Dr. Ir. C. Maathuis

INHOUDSOPGAVE

FIGUREN EN TABELLEN LIJST

VOORWOORD

Toen ik begon aan mijn afstudeertraject was de keuze voor de richting snel gemaakt: cybercriminaliteit. Het is een vakgebied dat mij intrigeert en waar tegenwoordig helaas iedereen mee in aanraking komt. Of dat nu is omdat men zelf gehackt wordt of dat men geen pinbetaling kan doen omdat de bank aangevallen is. Cybercriminaliteit is uiteraard zeer breed. Tijdens mijn oriëntatie kwam ik bij het onderzoekswerk van mijn begeleider Harald Vranken uit, onderzoek naar botnetdetectie. Dit onderzoek had raakvlakken met mijn eigen situatie: mijn bedrijf DDFR is een regionale Internet Service Provider (ISP). Als ISP hebben wij te maken met botnets. Aan de ene kant omdat botnets DDoS aanvallen uitvoeren op ons als geheel of specifiek op onze klanten en aan de andere kant omdat onze klanten geïnfecteerd kunnen zijn en hierdoor onbewust onderdeel zijn geworden van een botnet. Naast dat het onderzoek van waarde kan zijn voor DDFR kan ik op deze manier ook een bescheiden bijdrage leveren aan de bestrijding van cybercriminaliteit.

Deze scriptie is geschreven ter afronding van mijn studie Computer Science aan de Open Universiteit.

Een afstudeeronderzoek doe je niet alleen. Tijdens het project hebben vele mensen mij geholpen door hun kennis met mij te delen, hiervoor mijn dank. Een ieder persoonlijk bedanken is lastig, maar een aantal mensen wil ik toch noemen.
Als eerste mijn begeleider Harald Vranken, voor zijn begeleiding en feedback. Maar ook voor het begrip wanneer studie en werk in bepaalde perioden niet te combineren was.
Wesley en Erwin van der Bos, voor het ondersteunen bij en het maken van scripts voor het converteren van de ruwe data naar een bruikbare dataset. Een hels karwei!
Wouter de Vries van Tesorion voor het leveren van de input voor het labelen van de dataset. Een belangrijke stap in het hele proces.
Gerlof Dijk voor het nader toelichten van het DNS systeem en Nicolette Honsbeek, die mij de tijd en ruimte gaf om echt focus aan te brengen en een eerste versie wilde lezen.
D. Flud van Giffen voor de begeleidingsgesprekken, waarbij ik telkens uitgedaagd werd om tot de kern te komen van wat ik daadwerkelijk wilde vertellen.

Last but not least, mijn gezin, veel dank voor het geduld op momenten dat ik wel aanwezig was maar met mijn hoofd ergens anders.

Else Maria van der Meulen

Juni, 2021

SUMMARY

Botnets are networks of infected computers that can be remotely controlled and used for criminal purposes. They are sometimes being referred to as the Swiss pocket knife of cybercrime. Botnets pose a serious threat to our society. Over the years they have expanded considerably and have been further developed, making discovery and dismantling increasingly difficult. One of the developments is the use of new technologies like fast-flux and domain-flux. These techniques enable the rapid change of IP-addresses or domain-names, thus making it difficult to detect the Command and Control server of the botnet. Like most other devices, botnets connect to the internet by using Domain Name System (DNS) servers. The use of techniques like fast-flux and domain-flux leaves traces behind in the log files of DNS servers. This is the basis of my research question: is DNS the Achilles heel of flux-botnets?

Techniques to detect botnets have also further developed over the past years. A commonly used technique nowadays is machine learning, a form of artificial intelligence.
In this thesis I discuss my research on how DNS data and machine learning can help identify traces of botnets which are using fast-flux and domain-flux.

In the first fase I studied the current state of affairs regarding botnet detection and the use of machine learning. In the next step, an experimental field study, I developed and tested a machine learning model able to detect flux-botnets.

From the literature survey it became clear that virtually every botnet today uses DNS and fast-flux and domain-flux techniques.
It also appeared that DNS-based botnet detection is increasingly being applied, with machine learning as one of the most commonly used techniques.
An important part of developing a machine learning model is the selection of a dataset to train the model. Unfortunately, it quickly became clear that publicly available datasets were not appropriate. They are outdated, do not contain the data I need, or contained data from a non-comparable environment.

In order to carry out my field study, I therefore had to construct my own dataset. For this I used the log files of the recursive DNS servers owned and operated by DDFR, a Dutch internet service provider. This data was then enriched with GEO information and age of requested domain names, among others.
Machine learning models proved to work best if they are offered balanced data. In practice, especially with botnet detection, the data is not balanced. The amount of data related to malicious traffic is very small compared to bona fide traffic. For balancing datasets techniques such as undersampling (reducing the majority class) and oversampling (increasing the minority class) can be applied to balance a dataset, among others.

I performed the experiments with various machine learning algorithms and two datasets, one of which was unbalanced. With a balanced dataset and the Random Forest algorithm, my machine learning model provided a correct domain classification in 97.8% (Precision score). In 97,5% of the predictions a false classification is avoided (AUC score). The efficiency of the model to detect malicious domains is 97,5% (Recall score). When using an unbalanced test set these kind of scores are impossible.

Comparing my study to other research on botnet detection methods is difficult. The conditions under which experiments are carried out vary considerably. The application of evaluation methods is also not always uniform and results are not always known.

In my experiments I used 10 features. These include, among others, IP-addresses, their origin, where they are registered and age of the requested domain-names. Analysis of the training data showed that not all features are equally important and / or relevant.

My research has added a new way of botnet detection to the field. However, it remains unclear how various botnet detection techniques work in practice. I did not find any results regarding this during my literature survey. From the literature as well as my own research it has become clear that machine learning algorithms cannot cope with unbalanced data. There is a discrepancy between the form of data that enables machine learning algorithms to work best (balanced) and the form that occurs in reality (extremely unbalanced).

Further study of how theoretical methods of botnet detection work in practice is desirable. This also may provide new possibilities for practical applications sooner.

Finally, my research focused on using DNS data. Techniques like DoH (DNS over HTTPs) and DoT (DNS over TLS) are being used more and more by botnets. With these techniques they protect their communication with DNS servers and that has an implication on the botnet detection. The impact of this and how to deal with it, needs further investigation.

SAMENVATTING

Botnets zijn netwerken van geïnfecteerde computers die op afstand bediend en ingezet worden voor criminele doeleinden, zij worden ook wel eens het Zwitserse zakmes van de cybercriminaliteit genoemd. Botnets vormen een serieuze dreiging voor onze samenleving. Ze zijn door de jaren heen steeds groter geworden en steeds verder ontwikkeld, ontdekking en ontmanteling wordt hierdoor steeds lastiger.
Een van deze ontwikkelingen is dat botnets gebruik maken van fast-flux en domain-flux, technieken waarmee snel IP-adressen of domeinnamen gewisseld kunnen worden. De server die een botnet aanstuurt, ook wel Command & Control server genoemd, is daardoor moeilijker te ontdekken.
Net als elk ander apparaat, maken ook botnets verbinding met het internet met behulp van Domain Name System (DNS) servers. Het gebruik van technieken als fast-flux en domain-flux laat sporen achter in de logging van deze servers. Dit heeft geleid tot mijn onderzoeksvraag: is DNS de achilleshiel van flux-botnets?

Botnetdetectie technieken hebben zich door de jaren heen eveneens ontwikkeld. Een veel gebruikte techniek tegenwoordig is machine learning, een toepassingsgebied binnen kunstmatige intelligentie.
In deze scriptie wordt verslag gedaan van mijn onderzoek naar de wijze waarop DNS data en machine learning kunnen helpen bij het herkennen van sporen van botnets die fast-flux en domain-flux gebruiken.

In de eerste fase heb ik, via literatuurstudie, de huidige stand van zaken met betrekking tot botnetdetectie en machine learning onderzocht. Vervolgens is in de experimentele veldstudie een machine learning model ontworpen en getest waarmee flux-botnets gedetecteerd kunnen worden.

Uit het literatuuronderzoek werd duidelijk dat praktisch elk botnet tegenwoordig gebruik maakt van DNS en van fast-flux en domain-flux technieken.
Ook bleek dat botnetdetectie op basis van DNS steeds meer wordt toegepast. Hierbij is machine learning een geheel van technieken dat veel wordt toegepast.
Een belangrijk onderdeel bij het opzetten van een machine learning model is het kiezen van de dataset waarmee het model wordt getraind. Het werd snel duidelijk dat de publiekelijk beschikbare datasets niet passend waren. Datasets zijn verouderd, bevatten niet de data die ik nodig heb, of de data komt uit een niet vergelijkbare omgeving.

Derhalve, heb ik voor het uitvoeren van mijn veldstudie een eigen dataset opgebouwd op basis van logfiles van de recursive DNS-servers van DDFR, een kleine lokale Internet Service Provider. Deze data is vervolgens verrijkt met onder andere GEO informatie en ouderdom van opgevraagde domeinnamen.
Machine learning modellen bleken het beste te werken als de data die ze aangeboden krijgen, gebalanceerd is. In de praktijk, zeker bij botnetdetectie, is dit niet het geval. Het aandeel malafide verkeer is in verhouding met bonafide verkeer vaak zeer klein. Voor het balanceren van een dataset worden onder andere technieken als undersampling (verminderen van de meerderheidsklasse) en oversampling (vermeerderen van de minderheidsklasse) toegepast om een dataset te balanceren.

De experimenten heb ik uitgevoerd met diverse machine learning algoritmen en twee datasets, waarbij één dataset ongebalanceerde was. Bij een gebalanceerde dataset en gebruik makend van het algoritme Random Forest, gaf mijn machine learning model bij 97.8% een juiste classificatie van het domein (Precision score). In 97.5% van de voorspellingen wordt valse classificatie vermeden (AUC score). De effectiviteit van het model om malafide domeinen te detecteren is

97.5% (Recall score). Bij gebruik van een ongebalanceerde testset worden deze scores bij lange na niet gehaald.

Uit mijn literatuuronderzoek bleek dat vergelijken met andere detectiemethoden lastig is omdat de omstandigheden waaronder experimenten zijn uitgevoerd, verschillend zijn. De manier waarop evaluatiemethoden worden toegepast zijn niet altijd uniform terwijl de resultaten ook niet altijd bekend zijn.

Bij mijn experimenten heb ik tien features gebruikt. Deze hebben onder andere betrekking op IP-adressen, waar deze vandaan komen, waar deze geregistreerd zijn en de ouderdom van opgevraagde domeinnamen. Uit analyse van de trainingsdata komt naar voren dat niet alle features even relevant zijn.

Met mijn onderzoek is een nieuwe manier van botnetdetectie toegevoegd aan het onderzoeksveld. Wel blijft het onduidelijk hoe diverse botnetdetectie technieken in de praktijk werken. Hier heb ik tijdens mijn literatuuronderzoek geen resultaten voor gevonden. Uit de literatuur en ook uit mijn eigen onderzoek, blijkt dat machine learning algoritmen niet overweg kunnen met ongebalanceerde data. Er is dan ook een discrepantie tussen de vorm van de data waarmee machine learning algoritmen het beste kunnen werken (gebalanceerd) en de vorm die in de praktijk voorkomt (extreem ongebalanceerd).

Nader onderzoek is gewenst van hoe theoretische botnetdetectie methoden zich in de praktijk manifesteren. Hierdoor kunnen nieuwe mogelijkheden ook sneller een weg vinden naar toepassingen in de praktijk.

Tot slot, mijn onderzoek is gericht op het gebruik van DNS data. Technieken als DoH (DNS over Https) en DoT (DNS over TLS) worden meer en meer toegepast door botnets. Hiermee beveiligen zij hun communicatie met DNS servers, hetgeen gevolgen heeft voor het detecteren van deze botnets. Wat de impact zal zijn en hoe hier mee om te gaan, moet nog verder onderzocht worden.

TERMINOLOGIELIJST

ASN	AS = Autonomous System en betekent een groep IP-netwerken die wordt beheerd door een partij. Ook wel een routeringsdomein genoemd. Elk AS heeft een eigen nummer: het Autonomous System Number.
B-to-B klanten	Klanten uit de zakelijke markt. Geen consumenten.
Botnets	Netwerken gevormd door slave-host computers, bots genaamd, die worden bestuurd door een of meer aanvallers met de bedoeling kwaadwillende activiteiten uit te voeren (Silva, Silva, Pinto, & Salles, 2012).
Botmaster	Malafide gebruiker(s) die een botnet beheert en bestuurt.
Cache geheugen DNS server	Wanneer een DNS server een verzoek voor een domeinnaam heeft afgehandeld, wordt het antwoord tijdelijk opgeslagen. Mocht de domeinnaam weer opgevraagd worden, dan kan de DNS server snel antwoord geven. Dit wordt ook wel DNS Caching genoemd. Een timer, TTL genaamd, geeft aan hoelang een antwoord bewaard mag blijven.
Categorische variabele	Een variabele die wordt gemeten op de nominale schaal of ordinale schaal.
C&C	Command en Control.
C&C server	Server die commando's verstuurt naar de bots en waar de bots opdrachten / instructies vandaan halen.
Cybercrime	Alle delicten die gepleegd worden met behulp van ICT. Cybercrime omvat criminaliteit die is gericht op een ICT-systeem of waarvan de informatie door ICT wordt verwerkt.
Darkweb	Internetpagina's die worden gebruikt voor criminele doeleinden en die niet identificeerbaar zijn voor zoekmachines.
DGA	Domain Generation Algorithm, algoritme waarmee door botnets geautomatiseerd domeinnamen kan worden gegenereerd.
DoS / DDoS	Denial-of-Service / Distributed-Denial-of Service. DoS/DDoS aanvallen zijn pogingen om een computer / computernetwerk of dienst welke via het internet ontsloten is niet of moeilijk bereikbaar te maken voor de bedoelde klanten.
Deepweb	Internetpagina's die niet identificeerbaar zijn voor zoekmachines maar wel toegankelijk zijn met de juiste software. Darkweb is een onderdeel van het deepweb.
DNS	Domain Name System, het systeem en netwerkprotocol dat op het internet gebruikt wordt om namen van apparaten die aan internet zijn verbonden naar IP-adressen te vertalen en omgekeerd.

DoH	DNS over HTTPS, techniek waarmee communicatie met een DNS server wordt beveiligd.
DoT	DNS over TLS, techniek waarmee communicatie met een DNS server wordt beveiligd.
FQDN	Fully Qualified Domain Name.
GEO informatie	Geografische informatie m.b.t. IP-adressen. Dit betreft informatie in welk land een IP-adres is geregistreerd.
Hyperparameters	Parameters waarmee het model aangepast kan worden. Hyperparameters worden expliciet door de gebruiker gedefinieerd en worden vooraf toegepast voordat een machine learning algoritme wordt toegepast op een gegevens set. Er zijn hyperparameters om het model te optimaliseren en hyperparameters die model-specifiek zijn.
IDS	Intrusion Detection System: een geautomatiseerd systeem dat niet legitieme toegang tot informatiesystemen of netwerken ontdekt.
Internet Service Providers	Bedrijven die diensten leveren op of via het internet, waaronder het toegang verlenen tot internet. Dit kan zowel voor het bedrijfslevens als voor consumenten doeleinden zijn.
NXDomains	Antwoord van een DNS server wanneer een domein wordt opgevraagd dat niet bestaat.
P2P-protocollen	P2P-protocol = Peer-to-Peer protocol. Communicatie protocol in een netwerk waarbij aangesloten devices zoals bijvoorbeeld computers, servers, etc, elkaars gelijke zijn. Peer is Engels voor “gelijke”.
Wireshark	Computerprogramma welke gebruikt wordt om gegevens op een computernetwerk op te vangen en te analyseren. Het wordt ook wel een packet sniffer of een protocol analyzer genoemd.

1 INLEIDING

1.1 Maatschappelijke achtergrond

Cybercrime is een verschijnsel dat al bestaat sinds het begin van internet. De maatschappelijke invloed van cybercrime is door de jaren heen voortdurend groter geworden. Dit komt mede doordat onze samenleving steeds meer afhankelijk is geworden van internet. De Cybersecuritymonitor 2019 vermeldt dat 97.7% van alle Nederlanders vanaf 12 jaar, gebruik maakt van internet (CBS, 2019). Het CBS heeft ook het internetgebruik bij bedrijven onderzocht en concludeert dat bij alle bedrijven met 10 of meer medewerkers internet wordt gebruikt op regelmatige basis (CBS, 2021). Sinds 2020, ten tijde van de coronacrisis, is nog duidelijker naar voren gekomen hoe internet verweven is in onze maatschappij: thuiswerken, thuisonderwijs, zorg op afstand, bankzaken, pintransacties, online winkelen.

De Nationaal Coördinator Terrorismebestrijding en Veiligheid (NCTV) stelt dat cyberincidenten de maatschappij kunnen verlammen voor korte of langere tijd. Ook wordt vermeld dat cyberaanvallen een aantrekkelijk verdienmodel vormen voor criminelen (NCTV, 2020). Een voorbeeld hiervan is de Universiteit van Maastricht die €200.000 betaalde nadat ze was gehackt en gegijzeld met behulp van ransomware (NOS, 2020).

Het opzetten en gebruiken van botnets is één van de belangrijkste werktuigen van hackers en andere cybercriminelen. Een botnet is een netwerk van met malware geïnfecteerde computers, dat op afstand bediend en ingezet wordt voor cybercriminaliteit. Een botnet staat onder leiding van één of meerdere criminelen, de zogenaamde botmaster.
Botnets worden ingezet voor verschillende toepassingen. Bijvoorbeeld voor het uitvoeren van Distributed-Denial-of Service (DDoS) aanvallen. De stichting Nationale Beheersorganisatie Internet Providers (NBIP) constateert dat er in het derde kwartaal van 2020 meer DDoS aanvallen zijn uitgevoerd dan in het eerste kwartaal van 2020. Deze aanvallen waren krachtiger en vooral gericht op Internet Service Providers (ISP's) en grotere bedrijven en instellingen (NBIP, 2020). Het NCTV bevestigt dit beeld (NCTV, 2020).
Daarnaast worden botnets ook veelvuldig ingezet voor het verspreiden van nepnieuws waarmee de publieke opinie als ook democratische processen worden beïnvloed (Nieuwsuur, 2017), (Modderkolk, 2017). Andere activiteiten waarvoor botnets ingezet worden zijn: phishing aanvallen, spamaanvallen, malware, ransomware verspreiding en wachtwoord hacking. Met recht worden botnets wel het Zwitserse zakmes binnen cybercrime genoemd.

De groei in botnetactiviteiten was in het eerste half jaar van 2019 substantieel (ENISA, 2020). Botnets vormen een steeds grotere dreiging door hun groeiende omvang alsook door de continue verbeterende technieken om botnets te verbergen. Botnets zijn steeds geavanceerder geworden en worden gezien als de ruggengraat van cybercriminaliteit (Singh, Singh, & Kaur, 2019), (Asghari, Van Eeten, & Bauer, 2015).

1.2 Botnetdetectie

Parallel aan de ontwikkeling en groei van botnets is de detectie hiervan ontwikkeld. Er is veel onderzoek gedaan naar dit onderwerp en hiervan zijn meerdere overzichtsstudies gemaakt: (Silva, Silva, Pinto, & Salles, 2012), (Mahmoed, Nir, & Matrawy, 2014), (Garcia, Zunino, & Campo, 2014), (Acarali, Rajarajan, Komninos, & Herwono, 2016).

Eén van de detectiemethodes is het detecteren van botnets op basis van het Domain Name System (DNS). DNS is de naam van het systeem en het netwerkprotocol, dat gebruikt wordt om namen van apparaten die met internet zijn verbonden, naar IP-adressen te vertalen en omgekeerd.

Het aantal artikelen met een overzichtsstudie van DNS gerelateerde detectiemethoden is beperkter (Alieyan, Almomani, Manasrah, & Kadhum, 2017), (Li, Wang, & Zhang, 2017), (Singh, Singh, & Kaur, 2019). Veelal zijn de onderzoeken gericht op detectie op basis van Intrusion Dection Systemen (IDS), een geautomatiseerd systeem dat niet-legitieme toegang tot informatiesystemen of netwerken ontdekt, en in mindere mate gericht op DNS. In het laatste geval is er vooral onderzoek gedaan naar detectie op basis van domain-flux kenmerken. Domain-flux is het snel wisselen van domeinnamen behorend bij een IP-adres. In mindere mate is er ook onderzoek gedaan naar botnets die gebruik maken van fast-flux. Bij fast-flux worden de IP-adressen welke behoren bij een domeinnaam in hoog tempo gewisseld.
Veel botnets maken gebruik van het DNS en daarom is onderzoek hiernaar een welkome aanvulling in het detectielandschap.

Relatief weinig onderzoeken gebruiken data van provider netwerken en/of 'real-life' omgevingen (Stevanovic & Pedersen, 2013), (Garcia, Zunino, & Campo, 2014). Dit kan te maken hebben met het feit dat er weinig grootschalige real-life DNS datasets beschikbaar zijn (Zhauniarovich, Khalil, Yu, & Dacier, 2018), (Singh, Singh, & Kaur, 2019). Bovendien is van veel detectiemethoden niet bekend of deze ook daadwerkelijk zijn ingezet bij data afkomstig van bijvoorbeeld ISP's (Almomani, 2018), (Bilge, Kirda, Kruegel, & Balduzzi, 2011), (Bilge, Sen, Balzarotti, Kirda, & Kruegel, 2014), (Khattak, Ahmed, Syed, & Khayam, 2015).

Naast het feit dat publiekelijk beschikbare datasets lastig te verkrijgen zijn, zijn deze datasets ook relatief oud en daarmee mogelijk niet meer geschikt. Een ander probleem met verouderde datasets is dat wanneer men de data wil verrijken, dit niet meer mogelijk is omdat deze data niet meer voorhanden zijn. Bijvoorbeeld omdat het domein niet meer bestaat of omdat IP-adressen al overgedragen zijn of niet meer gebruikt worden.

1.3 Aanleiding onderzoek

Binnen mijn eigen bedrijf, DDFR genaamd, zie ik dagelijks de problematiek rondom botnets terug komen. DDFR is onder meer een Internet Service Provider (ISP), een partij die toegang verzorgt tot het internet. Dit kan specifiek zijn voor de consumentenmarkt of de zakelijke markt, of voor beide. Wij zijn specifiek gericht op de zakelijke markt en kunnen gekenmerkt worden als een kleine regionale ISP. Binnen DDFR kunnen klanten geïnfecteerd zijn en daarmee onderdeel geworden zijn van een botnet, ook hebben we te maken met aanvallen van botnets. Het aantal DDoS aanvallen is de afgelopen jaren steeds meer toegenomen. Niet alleen in aantal maar ook in omvang en complexiteit van de aanval.
Bij het zoeken naar een geschikt onderwerp voor mijn afstudeerscriptie, heb ik dan ook niet lang hoeven nadenken. Met botnets heb ik dagelijks te maken in mijn werk. Het doen van onderzoek naar dit onderwerp levert dan niet alleen een bescheiden bijdrage aan de onderzoekswereld op, maar mogelijk ook een verbetering voor mijn eigen bedrijf. Namelijk, het kunnen herkennen van geïnfecteerde klanten en hen informeren en ondersteunen om deze infectie te bestrijden.

De druk op ISP's om actie te ondernemen als klanten geïnfecteerd zijn, wordt de laatste tijd steeds groter. Deze acties kunnen variëren van het op de hoogte brengen van klanten als ze geïnfecteerd zijn, tot het uitvoeren van opschoonacties of het plaatsen van klanten in een quarantaine omgeving. In diverse onderzoeken worden Internet Service Providers aangewezen als de partij die verschil kan maken in het voorkomen en opschonen van botnetbesmettingen (Asghari, Van Eeten, & Bauer, 2015), (Jhaveri, Cetin, Gañán, & Van Eeten, 2017), (van Eeten, Bauer, Asghari, Tabatabaie, & Rand, 2010). Echter, een ISP is niet verantwoordelijk voor een eindgebruiker die zijn apparaten heeft laten infecteren, bijvoorbeeld door het niet uitvoeren van updates of door het gebruik van slechte beveiligde apparaten. Wel heeft een ISP de mogelijkheid om de communicatie van een botnet binnen zijn netwerk (deels) lam te leggen. Om dit te kunnen doen, moeten botnets wel eerst ontdekt en in kaart gebracht worden.

Om botnets op te sporen is data nodig. Het verkrijgen van data die geschikt is voor het doel van het onderzoek, kan in de praktijk erg lastig zijn. Echter, dankzij DDFR heb ik de mogelijkheid om eigen data te gebruiken. Voordeel hiervan is dat ik de uitkomsten ook daadwerkelijk kan toepassen binnen mijn bedrijf. Wanneer een externe dataset wordt gebruikt voor het onderzoek die niet afkomstig is van een ISP, dan zijn de uitkomsten niet zomaar te vertalen naar mijn situatie. De kenmerken van het internetverkeer van een ISP zijn anders dan het internetverkeer van bijvoorbeeld een hogeschool. Alleen al het feit dat de gebruikers uit andere doelgroepen bestaan en daarmee een ander verkeersbeeld genereren, heeft als gevolg dat deze kenmerken niet zomaar te vergelijken zijn.

Om duidelijk te maken welke data in mijn bedrijf geschikt is voor mijn onderzoek, licht ik eerst de werking van het Domain Name System toe. Elk apparaat dat verbonden is met het internet wordt geïdentificeerd met een IP-adres. Omdat IP-adressen lastig te onthouden zijn, is in de meeste gevallen een domeinnaam gekoppeld aan een IP-adres. Het opzoeken van een IP-adres dat bij een domeinnaam behoort, gebeurd door een Domain Name System (DNS). Er is echter niet één systeem op de wereld waarin alle IP-adressen met de bijbehorende domeinnamen zijn vastgelegd. Het systeem bestaat uit een verzameling van meerdere DNS servers die, via een boomstructuur, met elkaar in verbinding staan. Er zijn twee soorten servers: Authoritive Name Server en de Recursive Name Server. De Authoritive Name Servers zijn te vergelijken met telefoonboeken met daarin alleen netnummers die een regio beslaan zoals een ISP, een land, een specifiek gebied, etc. De Recursive Name Server is te vergelijken met een persoon die in een telefoonboek een nummer (IP-adres) opzoekt van een persoon (domeinnaam), of visa versa. Indien de Recursive Name Server het IP-adres niet weet, wordt het adres hoger in de DNS hiërarchie opgevraagd: de Authoritive Name Server.
Ik gebruik voor mijn onderzoek alleen de data die via onze Recursive Name Servers wordt afgehandeld. Wanneer in de rest van deze scriptie gesproken wordt over onze DNS servers wordt hiermee bedoeld onze Recursive Name Servers.

1.4 Probleemstelling

Botnets maken allemaal gebruik van internet en de DNS structuur is de basis van internet. Domain-flux en fast-flux zijn tegenwoordig veel gebruikte technieken om een botnet te beschermen (Singh, Singh, & Kaur, 2019), (Alieyan, Almomani, Manasrah, & Kadhum, 2017). De wereld van soorten botnets is dermate groot dat ik mij beperk tot botnets die gebruik maken van fast-flux en/of domain-flux: flux-botnets.

Het is een kat en muis spel: botnets ontwikkelen en gebruiken steeds nieuwe technieken om ontdekking en ontmanteling te voorkomen en botnetdetectie onderzoek richt zich vervolgens op nieuwe detectiemethoden. Een techniek die bij de detectiemethoden veel wordt toegepast is machine learning, een toepassingsgebied binnen de kunstmatige intelligentie. Machine learning omvat meerdere leertechnieken voor de ontwikkeling van intelligente systemen. Deze systemen leren op basis van aangeboden data en kunnen zich verbeteren, zonder hiervoor expliciet geprogrammeerd te worden. Wanneer flux-botnets actief zijn, zouden die patronen moeten achterlaten in de data: zou dan DNS de achilleshiel van flux-botnets zijn?

1.5 Onderzoeksvraag

In het kader van de in bovenstaande paragrafen beschreven problematiek, kom ik tot de volgende centrale onderzoeksvraag voor mijn eindscriptie:
Hoe kunnen botnets die gebruik maken van fast-flux en domain-flux technieken, gedetecteerd worden met behulp van machine learning, waarbij gebruik gemaakt wordt van recursieve DNS data?

Om antwoord te kunnen geven op deze onderzoeksvraag zullen de volgende deelvragen beantwoord moeten worden:

1. *Wat zijn botnets en hoe werken ze?*
2. *Hoe werken de technieken fast-flux en domain-flux en hoe worden ze toegepast? Zijn er overeenkomende kenmerken tussen deze twee technieken?*
3. *Welke taxonomie wordt gebruikt voor botnetdetectie op basis van DNS verkeer en welke machine learning algoritmen worden hierbij vooral toegepast?*
4. *Wat zijn de belangrijkste recursive DNS kenmerken wanneer botnets gebruik maken van fast-flux en domain-flux? Welke features kunnen hieruit worden geselecteerd?*
5. *Welke dataset kan gebruikt worden en hoe kan de ground truth hiervan bepaald worden?*
6. *Welk machine learning algoritme kan hierbij gebruikt worden? Wat is de succesratio van dit algoritme en hoe staat dit in verhouding met andere, reeds bekende, oplossingen?*

1.6 Leeswijzer

In hoofdstuk 2 wordt de wijze waarop het onderzoek is uitgevoerd beschreven.

Hoofdstuk 3 geeft meer informatie over botnets en specifiek over fast-flux en domain-flux botnets. Ook wordt beschreven welke verschillende botnetdetectie methoden er bestaan en waar DNS gebaseerde technieken onder vallen. Tot slot wordt toegelicht welke onderverdeling binnen de op DNS gebaseerde methoden gemaakt kan worden.
Hiermee worden de deelvragen 1 en 2 beantwoord.

Hoofdstuk 4 gaat in op machine learning, de achtergrond en indeling van machine learning en specifiek hoe deze technieken toegepast worden bij onderzoek dat op DNS gebaseerd is. Tot slot worden een aantal algoritmen nader toegelicht.

Hoofdstuk 5 geeft een overzicht van binnen de literatuur bekende datasets die toegepast zijn bij botnetdetectie op basis van DNS. Daarna wordt beschreven hoe de eigen dataset is opgezet, welke kenmerken (features) zijn geselecteerd en hoe labeling, dat wil zeggen het proces waarbij gegevens worden geïdentificeerd en voorzien van een label met zinvolle informatie, is uitgevoerd. Aan het einde worden problemen besproken waar ik tegenaan gelopen ben bij het opzetten en labelen van de dataset.

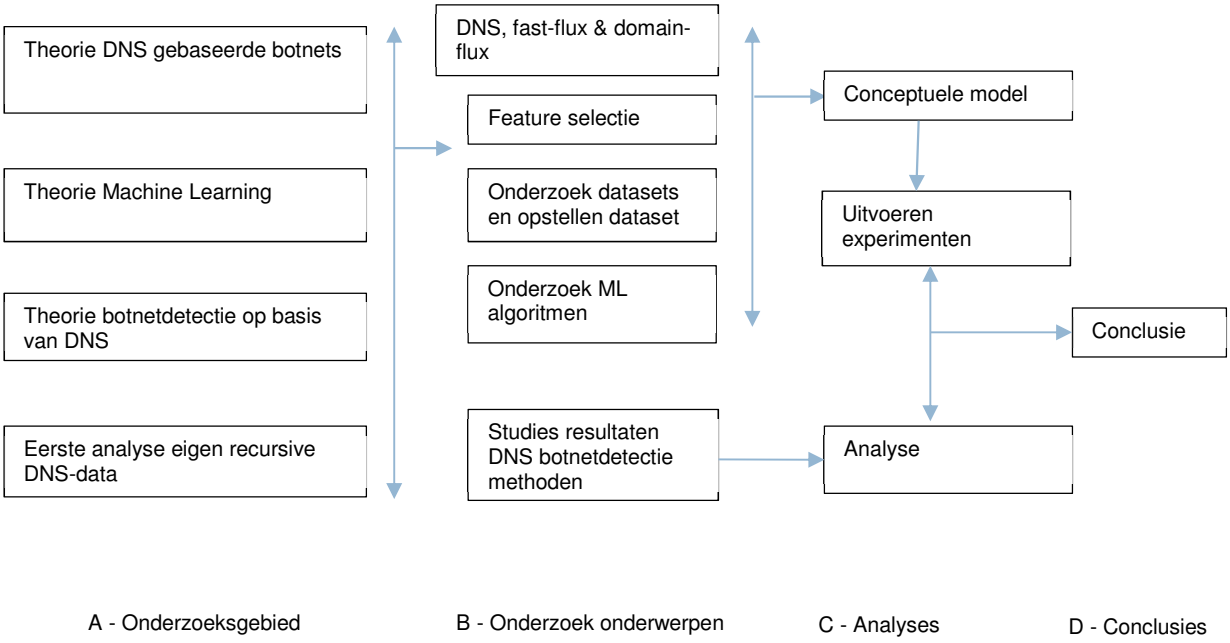
Hoofdstuk 6 beschrijft de opzet en uitvoering van de experimenten. Uitkomsten worden vergeleken met resultaten van andere op DNS gebaseerde botdetectie methoden.

Tot slot wordt in hoofdstuk 7 de onderzoeksvraag beantwoord en zijn conclusies en discussies naar aanleiding van het onderzoek te vinden. Toegelicht wordt de bijdrage van het onderzoek aan het onderzoeksveld. Ook worden suggesties gedaan voor nader onderzoek. Dit hoofdstuk eindigt met mijn reflectie over het onderzoekstraject.

2 ONDERZOEKSMETHODE

Het onderzoek dat ik voor deze eindopdracht uitgevoerd heb, kan worden gekenmerkt als een exploratief-descriptief onderzoek. Enerzijds is, aan de hand van literatuurstudie, de huidige stand van zaken rondom botnetdetectie in kaart gebracht, alsmede onderzocht op welke manier machine learning daarbij ingezet wordt. Anderzijds is, in een experimentele veldstudie, onderzocht hoe flux-botnets gedetecteerd kunnen worden met behulp van machine learning. Gaandeweg is in het onderzoek vorm en inhoud gegeven aan hoe deze detectie dan tot stand moet komen.

Het onderzoek kan beschreven worden met behulp van het model van Doorewaard en Verschuren (Doorewaard & Verschuren, 2015). Dit model geeft globaal de structuur weer en geeft inzicht in de uitgevoerde stappen en de samenhang daartussen.
In Figuur 1 worden deze stappen schematisch weergegeven.



FIGUUR 1 ONDERZOEK MODEL

2.1 Onderzoeksgebied

In fase A-Onderzoeksgebied, is een verkennend literatuuronderzoek gedaan naar botnets, machine learning en botnetdetectie op basis van DNS data. Ook is er een eerste analyse gedaan van de eigen DNS data om te kijken of deze data als input gebruikt kon worden voor het opzetten van een eigen dataset.

Voor dit literatuuronderzoek zijn de volgende digitale wetenschappelijke databases gebruikt: IEEE, ACM, Springer, Science Direct, Future internet.

Deze bronnen worden binnen de wetenschap als valide beschouwd. Daarnaast werd bij elk gevonden document gekeken naar onderstaande punten om de validiteit van het document te beoordelen:

- Is traceerbaar waar het document in is gepubliceerd;
- Is het document geciteerd en zo ja hoe vaak en is dit op betrouwbare wijze gedaan;
- Is het document gepubliceerd in een wetenschappelijk tijdschrift, boek of in conference proceedings, of kan op een andere wijze de wetenschappelijke achtergrond achterhaald worden.

- Ouderdom bron: bronnen die betrekking hebben op ontwikkelingen bij voorkeur niet ouder dan 5 jaar.

Op basis van dit literatuuronderzoek zijn de probleemstelling en de onderzoeksvraag geformuleerd en is een beeld gevormd van de achtergrond en de relevantie van het probleem (hoofdstuk 1).

2.2 Onderzoek onderwerpen

In Fase B-Onderzoek onderwerpen, is aanvullend literatuuronderzoek gedaan naar de volgende specifieke onderwerpen:

- Botnets, werking van DNS, fast-flux en domain-flux (hoofdstuk 3);
- Machine learning en specifiek gebruik machine learning bij detectie van botnets op basis van DNS (hoofdstuk 4);
- Gepubliceerde datasets en wijze waarop deze zijn beschreven (hoofdstuk 5).

In deze fase is ook onderzocht welke machine learning (ML) algoritmen van toepassing kunnen zijn. Tevens zijn de kenmerken (features) geselecteerd die als input dienen voor dit algoritme. De kenmerken zijn gekozen op basis van literatuuronderzoek waarbij ook is gekeken naar een combinatie van kenmerken die nog niet eerder is onderzocht.

Na het kiezen van de kenmerken is onderzoek gedaan naar bestaande datasets. Hierbij is met name de vraag aan de orde geweest in hoeverre deze passend zijn voor mijn onderzoek of dat het beter zou zijn een eigen dataset op te stellen. Conclusie was dat het voor mijn onderzoek nodig was een eigen dataset te creëren. Het samenstellen van een eigen dataset is gebeurd aan de hand van het framework van Ring et al. (Ring M. , Wunderlich, Gruedl, Landes, & Hotho, 2017). Dit framework is gebaseerd op de vier principes van Wilkinson (Wilkinson, 2016) waar wetenschappelijke data aan moet voldoen: vindbaarheid, toegankelijkheid, interoperabiliteit en herbruikbaarheid.

2.3 Analyses

In fase C-Analyses, zijn verschillende machine learning modellen opgezet, getraind en getest (hoofdstuk 6). De uitkomsten van de experimenten zijn geanalyseerd en geëvalueerd. De gebruikte methodes zijn beschreven in hoofdstuk 4 (paragraaf 4.6.2). De performance wordt vergeleken met fast-flux & domain-flux detectie methoden die in de literatuur beschreven zijn.

2.4 Conclusies

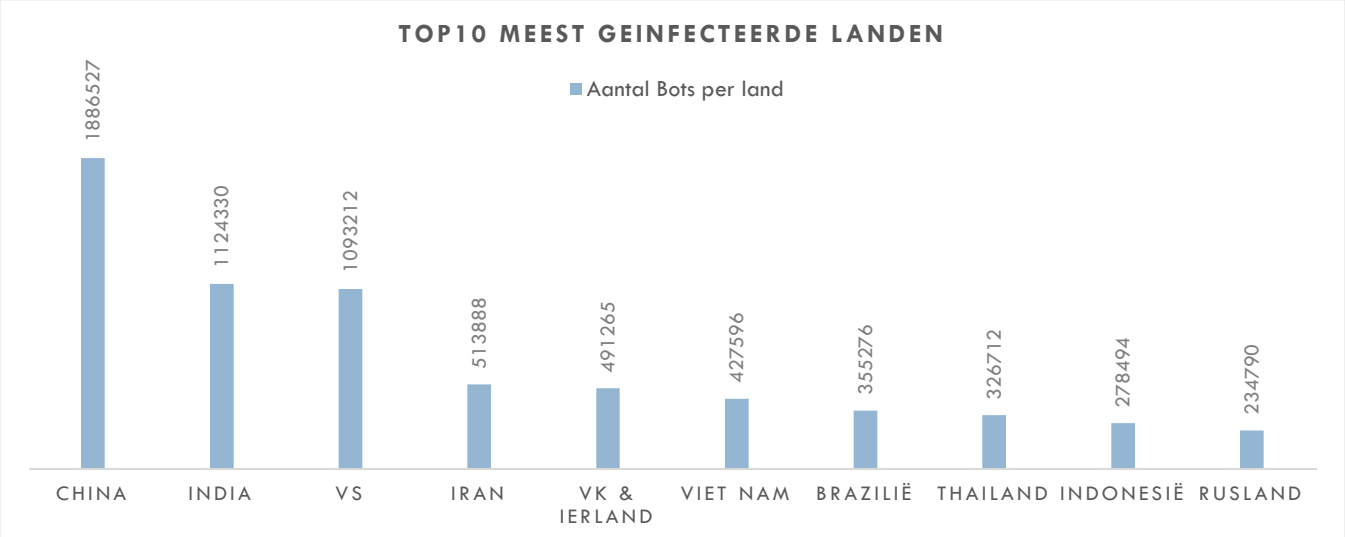
In Fase D-Conclusies, wordt aan de hand van de uitkomsten antwoord gegeven op de onderzoeksvraag en de deelvragen (hoofdstuk 7). Ook worden eventuele tekortkomingen en/of discussiepunten en suggesties voor nader onderzoek beschreven.

3 FAST-FLUX EN DOMAIN-FLUX BOTNETS

3.1 Introductie

Als het tegenwoordig over botnets gaat, dan wordt daarmee meestal verwezen naar netwerken met kwaadwillende bedoelingen. Zo definiëert S. Silva et al. de term als volgt: "Botnets are networks formed by 'enslaving' host computers, called bots (derived from the word robot), that are controlled by one or more attackers, called botmasters, with the intention of performing malicious activities." (Silva, Silva, Pinto, & Salles, 2012, p. 380). Echter, het ontstaan van botnets heeft niets te maken met kwaadwillende activiteiten.
Botnets zijn ontstaan vanuit een op tekst gebaseerd chatsysteem, het zogenaamde Internet Relay Chat (IRC). Gebruikers kunnen lid worden van een kanaal op een IRC-netwerk en zo communiceren met andere gebruikers of groepen. Hierbij werden bots ingezet om drukke kanalen te beheren. Bots zijn computerprogramma's die op autonome wijze taken kunnen uitvoeren die normaal door mensen worden uitgevoerd. Bots waren onder andere in staat om simpele opdrachten uit te voeren, administratieve ondersteuning te bieden, simpele spelletjes aan te bieden en informatie op te halen over bijvoorbeeld het operating systeem, emailadressen etc. (Silva, Silva, Pinto, & Salles, 2012). Uiteindelijk zijn bots uitgegroeid tot een uitgebreid hulpmiddel om een IRC-server te beheren. Een voorbeeld van zo'n hulpmiddel is Eggdrop, dit wordt beschouwd als het eerste, niet-kwaadwillende botnet, het werd gepubliceerd in 1993 (Silva, Silva, Pinto, & Salles, 2012), (Pointer, 2018). Eggdrop is een legitiem IRC-botnet dat nog steeds verder ontwikkeld wordt.

Tegenwoordig wordt met de aanduiding botnet een botnet bedoeld met kwaadwillende, illegale activiteiten. In de rest van mijn scriptie wordt ook deze betekenis gehanteerd, tenzij uitdrukkelijk anders aangegeven.
Op basis van Eggdrop zijn nieuwe IRC-botnets ontwikkeld maar nu met het doel om andere IRC-gebruikers en –servers aan te vallen. Het eerste bekende kwaadaardige botnet is GTBot dat verscheen rond 1998 (Silva, Silva, Pinto, & Salles, 2012). De eerste botnets werden vooral ontwikkeld door mensen die het leuk vonden iets illegaals te doen. Tegenwoordig is het ontwikkelen en gebruiken van botnets vooral een enorme professionele onderneming geworden. Om een idee te geven van de invloed van botnets is in Figuur 2 een overzicht gegeven van de top 10 landen met de meeste botnetinfecties gegeven. Deze lijst uit 2021, is afkomstig van het Spamhaus project dat onder andere onderzoek doet naar aantallen botnetinfecties per land (The Spamhaus project, 1998-2021).



FIGUUR 2 TOP 10 LANDEN MET MEESTE BOTNET INFECTIES

Volgens Silva et al. (Silva, Silva, Pinto, & Salles, 2012, pp. 381,382) bestaat elk botnet uit de volgende drie componenten:

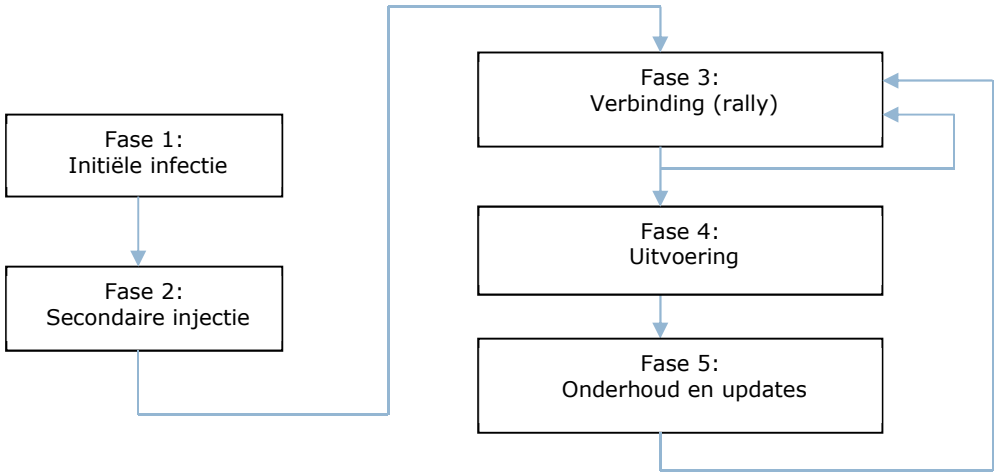
Botmaster/botherder
De botmaster, ook wel botherder genoemd, is een malafide gebruiker die het botnetwerk beheert en bestuurt. De botmaster geeft de commando's aan de bots om illegale activiteiten uit te voeren.

Bot/Slave
De slave is het apparaat waarop een softwareprogramma is geïnstalleerd waarmee de botmaster in staat is het apparaat te beheren. Apparaten kunnen op diverse manieren worden geïnfecteerd: via e-mail, geïnfecteerde websites of zwakheden in de programmatuur van het apparaat (lekken) waardoor infectie kan plaatsvinden. Een bot/slave is altijd onderdeel van een groter netwerk. Bovendien is een bot uitgerust met een communicatiekanaal waarlangs het bot opdrachten krijgt en communiceert met de botmaster.

Command en Control (C&C) server
Elk botnet heeft een infrastructuur waarlangs de commando's worden verstuurd naar de bots en waarmee gecommuniceerd wordt met de botmaster. De manier waarop deze communicatie-infrastructuur is georganiseerd onderscheidt de verschillende typen botnets (zie paragraaf 3.3).

3.2 Levenscycli botnets

Een botnet is geen statisch geheel maar doorloopt verschillende fasen in zijn bestaan. Hoewel in de literatuur de benamingen voor de fasen verschillend zijn en er een variatie is in het aantal fasen, komt de strekking op hetzelfde neer. Figuur 3 is een weergave van de fasen zoals benoemd door Silva et al. (Silva, Silva, Pinto, & Salles, 2012).



FIGUUR 3 DE LEVENSFASEN VAN EEN BOTNET (SILVA ET AL. 2013)

Fase 1, Initiële infectie: een bepaald apparaat wordt geïnfecteerd en wordt daarmee een potentieel bot. Het infecteren van deze zogenaamde host kan op verschillende manieren worden gedaan. Een voorbeeld hiervan is het versturen van e-mails met geïnfecteerde bestanden. Ook het uitbuiten van kwetsbaarheden in systemen die worden ontdekt door geautomatiseerde scanning wordt veel toegepast.

Fase 2, Secondaire injectie: wanneer de eerste fase succesvol is, wordt er een programma uitgevoerd dat op zoek gaat naar kwaadaardige software, zogenaamde malware binaries, die vervolgens worden gedownload en uitgevoerd.
Fase 3, Verbinding (rally): een bot/slave zoekt contact met de Command en Control (C&C) server voor instructies en/of updates. Dit proces van contact zoeken met de C&C-server wordt 'rallying' genoemd. Het bot zal periodiek contact blijven maken voor nieuwe instructies en/of updates. Het geïnfecteerde apparaat is nu een bot geworden en onderdeel van het botnetwerk.
Fase 4, Uitvoering: in deze fase is het botnetwerk klaar om aanvallen uit te voeren. Het doel van een aanval kan zeer divers zijn, bijvoorbeeld, het uitvoeren van een DDoS aanval, het stelen van gegevens, het opeisen van computerrekenkracht, spamaanval, etc.
Fase 5, Onderhoud en updates: een botnet zal onderhouden moeten worden, bijvoorbeeld om ontdekking te voorkomen of om nieuwe mogelijkheden aan het botnet toe te voegen.

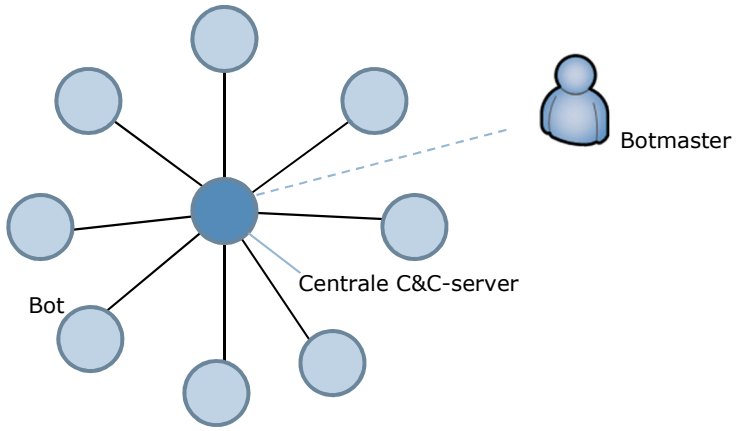
Deze fasen kunnen diverse keren doorlopen worden, immers, nieuwe bots worden continue toegevoegd aan het netwerk.

3.3 Infrastructuur van botnets

Botnets zijn door de jaren heen geëvolueerd tot netwerken met een steeds meer geavanceerde infrastructuur. Onder andere Silva et al. (Silva, Silva, Pinto, & Salles, 2012) en Hyslip & Pittman (Hyslip & Pittman, 2015) hebben hier onderzoek naar gedaan. De ontwikkeling van de infrastructuur heeft zich met name gericht op het meest kwetsbare onderdeel, namelijk de manier waarop bots met de Command & Control server communiceren.
Botnets kunnen geclassificeerd worden op basis van hoe de bots communiceren met de botmaster en visa versa, dat wil zeggen op basis van de infrastructuur van een botnet. Hieronder worden de drie meest bekende hiervan besproken.

3.3.1 Centrale C&C infrastructuur

In dit model gebruikt een botmaster één of meerdere centrale servers voor het aansturen van zijn bots. Deze infrastructuur is vergelijkbaar met het klassieke client-server netwerkmodel. Figuur 4 laat een weergave zien van een centrale C&C-infrastructuur.



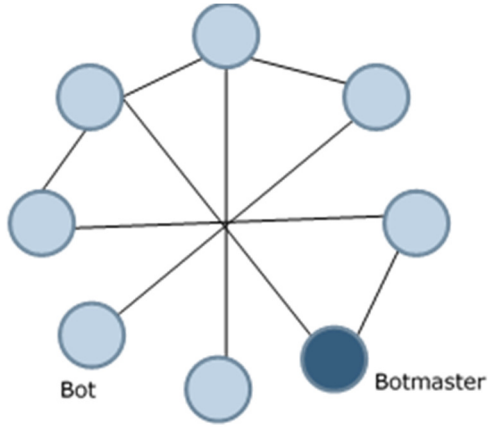
FIGUUR 4 CENTRALE C&C INFRASTRUCTUUR

Voordelen van de centrale C&C-infrastructuur zijn de eenvoudige implementatie, het simpele beheer en de lage latency. Tussen het versturen en ontvangen van pakketten over een netwerk zit altijd een x-hoeveelheid tijd. Deze vertraging in dataoverdracht over een netwerk wordt latency genoemd. Lage latency, weinig vertraging, is van belang bij het uitvoeren van bijvoorbeeld DDoS aanvallen. De eenvoud van de centrale C&C-infrastructuur is ook gelijk zijn zwakke plek. Het is

namelijk relatief eenvoudig om een dergelijke server te lokaliseren, uit te schakelen en daarmee het netwerk te ontmantelen.

3.3.2 Decentrale C&C infrastructuur

Als reactie op de zwakte van een centrale C&C infrastructuur werd een decentrale infrastructuur ontwikkeld. Hiermee is een botnet beter beschermd tegen ontdekking en ontmanteling. Dergelijke botnets worden ook wel Peer-2-Peer(P2P)-botnets genoemd omdat voor de communicatie zg. P2P-protocollen worden gebruikt. Het woord peer komt uit het Engels en betekent "gelijke". Bij de decentrale infrastructuur is een extra communicatie laag gecreëerd waarbinnen bots gelijkwaardig aan elkaar zijn: ze functioneren zowel als server en als cliënt. Wanneer een bot zich aanmeldt bij het netwerk wordt verbinding gemaakt met een aantal meegegeven bots (peers) in plaats van direct met de C&C server. Daarna worden de peers uitgewisseld voor het verbeteren van de onderlinge communicatie. In deze infrastructuur kunnen bots fungeren als opdrachtgever (commando's versturen) en ook als uitvoerder van opdrachten. De botmaster kan vervolgens via een willekeurige bot commando's versturen of informatie ophalen. Figuur 5 laat een weergave zien van een decentrale C&C-infrastructuur.

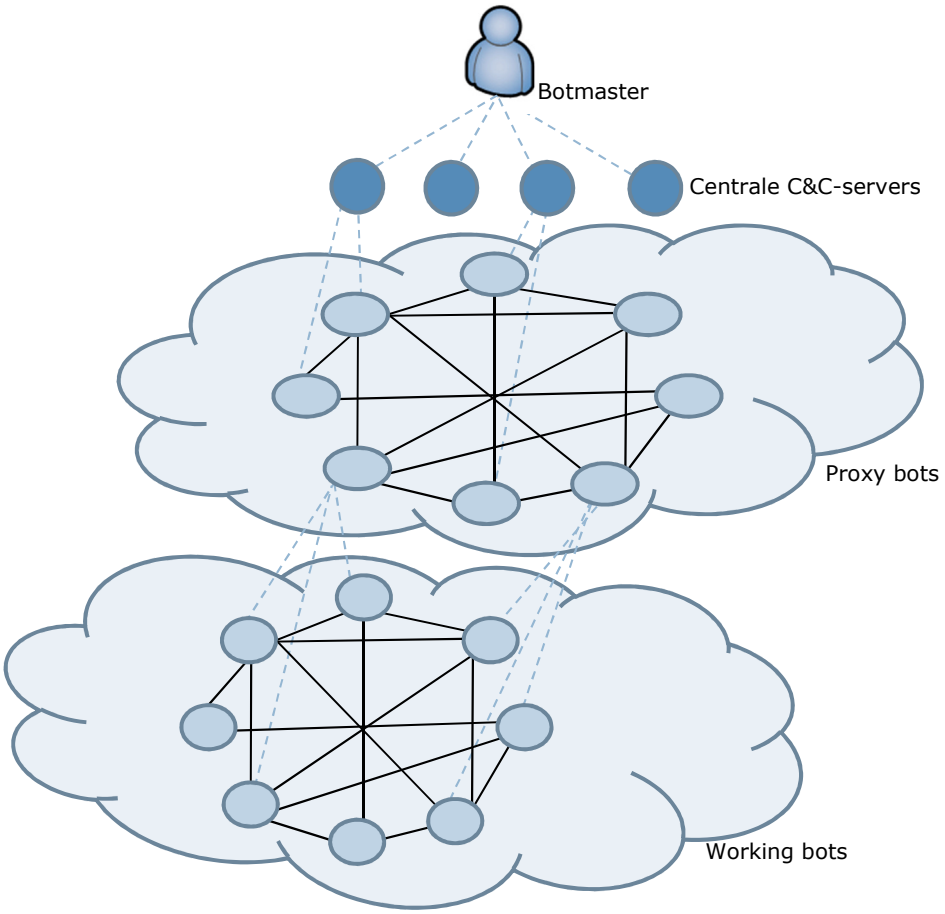


FIGUUR 5 DECENTRALE C&C INFRASTRUCTUUR / P2P-BOTNETWERKEN.

Een P2P-botnetwerk is aanzienlijk complexer dan een centraal aangestuurd botnetwerk. Daardoor is ontdekking en ontmanteling een stuk lastiger. Ook al wordt een deel ontmanteld, dan nog kan het botnetwerk zijn werk verrichten. Nadeel is de hoge latency van een P2P-botnetwerk als gevolg van het continue synchroniseren van de bots onderling. Dit beperkt de mogelijkheden van het botnetwerk om kwaadaardige acties uit te voeren, daardoor is dit netwerk minder geschikt voor het uitvoeren van DDoS aanvallen.

3.3.3 Hybride C&C infrastructuur

Om de voordelen van zowel een centrale als van een decentrale infrastructuur te kunnen benutten, is de hybride C&C-infrastructuur ontwikkeld. Hierbij worden beide infrastructuren gecombineerd zodat zowel snelheid gerealiseerd kan worden als ook bescherming tegen ontdekking en ontmanteling. In Figuur 6 is de hybride C&C-infrastructuur weergegeven.



FIGUUR 6 HYBRIDE INFRASTRUCTUUR

Binnen een hybride C&C-infrastructuur kunnen twee groepen bots worden onderscheiden: working bots, cliënts, die commando's uitvoeren en proxy bots, servers, die zowel commando's kunnen geven als uitvoeren. Working bots kunnen alleen optreden als cliënt. Een working bot heeft periodiek contact met proxy bots om commando's of updates te ontvangen. De proxy bots zorgen door middel van P2P communicatie voor de verspreiding van de commando's en updates.

3.3.4 Beveiligen van de C&C infrastructuur

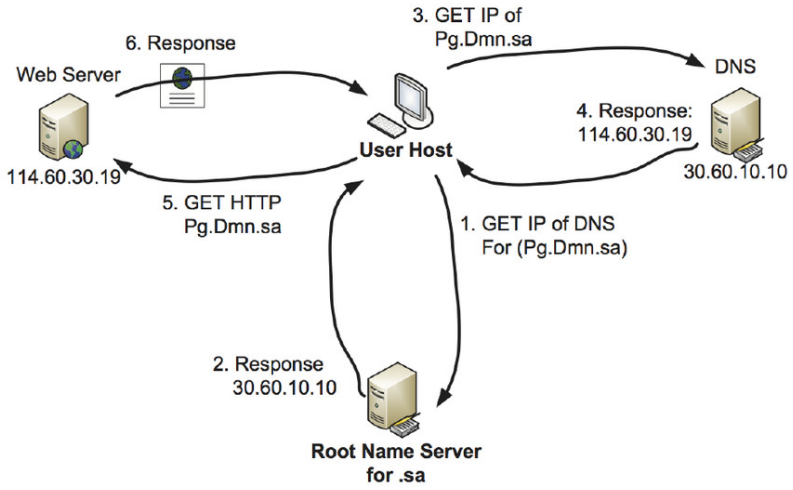
Naast het ontwikkelen van de Command & Control infrastructuur zijn ook andere technieken toegepast om beter beschermd te zijn tegen ontdekking en ontmanteling. Een van de eerste verbeteringen was het toepassen van encryptie. Alle moderne botnets passen tegenwoordig encryptie toe (Silva, Silva, Pinto, & Salles, 2012).
Ook de manier waarop in de verbinding fase (fase 3) contact wordt opgenomen met de C&C server is door de jaren heen veranderd (Singh, Singh, & Kaur, 2019). Bij de eerste botnets met een centrale infrastructuur werd vooral gebruik gemaakt van een vast IP-adres. Dit biedt echter weinig bescherming. Een vast IP-adres kan eenvoudig worden gedetecteerd met het gebruik van een 'honeypot'. Dit is een computersysteem verbonden met het internet, bewust onbeveiligd en dus kwetsbaar voor malware. Door een dergelijk computersysteem te laten infecteren, wordt interessante aanvalsinformatie verkregen.
Wanneer een gedetecteerd IP-adres op een zwarte lijst wordt gezet, de zogenaamde 'blacklist', kan het botnet (gedeeltelijk) lamgelegd worden. Om deze blacklist te omzeilen, gebruiken botnets tegenwoordig technieken als fast-flux en domain-flux. Deze technieken worden respectievelijk in paragraaf 3.5.1 en paragraaf 3.5.2 besproken.

Het is van belang te weten welke infrastructuren en beveiligingstechnieken gebruikt kunnen worden. Immers, elke infrastructuur en/of beveiligingstechniek laat bepaalde patronen zien in het

door botnets gegenereerde internetverkeer. Detectietechnieken kunnen worden ingezet om deze patronen op te sporen.

3.4 Domain Name System

Het Domain Name System (DNS) is een essentieel onderdeel van internet en wordt door elk apparaat gebruikt om verbinding te maken met internet. De werking van dit systeem is gebaseerd op een client-server model (IETF, 1987), (IETF, 1987). Een cliënt vraagt een domeinnaam of een IP-adres op bij een DNS-server en deze stuurt weer een antwoord terug. Indien de server het antwoord op de vraag niet weet, wordt de vraag doorgestuurd naar een server hogerop in de hiërarchie. Figuur 7 laat zien hoe informatie wordt opgevraagd bij een webserver in een normale situatie (Mahmoed, Nir, & Matrawy, 2014).



FIGUUR 7 NORMALE DNS AANVRAAG (M. MAHMOUD ET AL. 2014)

Uitleg: wanneer een gebruiker verbinding zoekt met de website genaamd PG.Dmn.sa. worden de volgende stappen uitgevoerd (Mahmoed, Nir, & Matrawy, 2014, p. 5):

Stap 1: 'user host' (gebruiker) vraagt bij de ".sa" root name server het IP-adres op van de DNS server die "Dmn.sa" host.
Stap 2: het IP-adres 30.60.10.10 wordt door de ".sa" root name server terug gestuurd.
Stap 3: vervolgens wordt door de user host het IP-adres 30.60.10.10 gebruikt om contact te maken met de betreffende DNS server en het IP-adres voor "Pg.Dmn.sa" op te vragen.
Stap 4: deze server geeft het IP-adres 114.60.30.19 retour.
Stap 5: de user host gebruikt het IP-adres 114.60.30.19 om deze webserver te benaderen en de gewenste informatie op te halen.
Stap 6: de webserver geeft de gevraagde informatie retour aan de user host.

3.4.1 DNS vraag en antwoord

Alle communicatie binnen het DNS protocol heeft hetzelfde format. Elk bericht wordt een 'message' genoemd en is onderverdeeld in vijf secties, te weten: Header, Question (opgevraagde domeinnaam), Answer, Authority en Additional (IETF, 1987).
De header-sectie, of kop, is altijd aanwezig en bevat onder andere informatie of het een query (vraag) of response (antwoord) message betreft. De segmenten Answer, Authority en Additional zijn alleen gevuld wanneer het een antwoord betreft. Is dit het geval dan kunnen deze segmenten één of meerdere keren voorkomen. Bijvoorbeeld, een domein met drie IP-adressen zal driemaal een Answer segment bevatten waarbij in elk segment de informatie zit van één van de drie IP-adressen. Hetzelfde geldt voor het aantal Authoritive NameServers: als voor een domeinnaam

meerdere domainservers ingesteld zijn, zullen er ook meerdere Authority-segmenten gevuld zijn. Zie Figuur 8 voor een toelichting van de verschillende onderdelen van een DNS message.

Header	Onderverdeling	Toelichting
Header	ID	ID wordt automatisch gegenereerd bij de vraag en gekopieerd bij het bijbehorende antwoord. Antwoorden kunnen worden zo gematched worden met openstaande vragen.
	Parameters (Flags)	Specificeert o.a. of het bericht een vraag of antwoord is en in geval van een vraag, wat voor soort vraag (recursive of non-recursive). Bij een antwoord komt ook een response code mee. Deze kan de waarde oké hebben of een van vijf verschillende foutmeldingen.
	QDCount	Geeft aan hoeveel entries er in het Question segment zitten.
	ANCount	Geeft aan hoeveel entries er in het Answer segment zitten.
	NSCount	Geeft aan hoeveel entries er in het Authority segment zitten. Bij een quer message staat het altijd op 0.
	ARCount	Geeft aan hoeveel entries er in het Additioneel segment zitten.
Question	QName	De opgevraagde domeinnaam.
	QType	Soort vraag bijv. hostadress (A Type) of een mail adres (MX type), etc.
	QClass	Class is in het algemeen ingesteld op IN (internet) voor de DNS-records met hostnamen, servers of IP-adressen.
Answer	Name	Volledig opgevraagde domeinnaam.
	Type	Recordtype. Geeft het format van de gegevens aan en een idee van het beoogde gebruik: MX-record specificeert de mailserver voor een opgegeven domeinnaam, een A-record de vertaling van domeinnaam naar IPv4 adres, AAAA-record vertaling van domeinnaam naar IPv6 adres, etc.
	Class	Class is in het algemeen ingesteld op IN (internet) voor de DNS-records met hostnamen, servers of IP-adressen.
	TTL	Time To Live: de tijd waarin het domein nog in het cache geheugen mag staan van een DNS server.
	RDLength	Geeft lengte van het Rdata veld aan.
	RData	De opgevraagde data, bijvoorbeeld IP-adres of een andere domeinnaam.
Authority	Name	Dit segment bevat dezelfde velden als het Answer-segment maar dan met andere waarden. In dit segment word(t)(en) de na(a)m(en) van de Authoritive NameServer(s) van het opgevraagde domein teruggegeven, indien aanwezig. In dat geval geeft het veld Name: de naam van het opgevraagde domein, type = NS (NameServer), class = IN, TTL= opgegeven tijd voordat record opnieuw moet worden opgevraagd, RDLength = lengte van het RData veld, RData = de naam van de NameServer(s). Indien aanwezig zal per Authoritive NameServer een Authority segment aanwezig zijn.
	Type	
	Class	
	TTL	
	RDLength	
	RData	
Additional	Name	Dit segment bevat dezelfde velden als het Answer-segment maar dan met andere waarden. In dit segment worden, indien aanwezig, de IP-adressen van de opgevraagde Authoritive NameServers teruggegeven. In dat geval geeft het veld Name: de naam van de NameServer, type = A, class = IN, TTL, RDLength = lengte van het RData veld, RData = het IP-adres van de opgevraagde NameServer. Per aanwezige Authoritive NameServer zal een Additional segment aanwezig zijn.
	Type	
	Class	
	TTL	
	RDLength	
	RData	

FIGUUR 8 DNS MESSAGE ONDERVERDEELD IN SEGMENTEN

In Figuur 9 hieronder is een voorbeeld te zien van het opvragen van de domeinnaam NOS.nl.

```
Domain Name System (query)
    Transaction ID: 0x8485
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.nos.nl: type A, class IN
          Name: www.nos.nl
          [Name Length: 10]
          [Label Count: 3]
          Type: A (Host Address) (1)
          Class: IN (0x0001)
    [Response In: 63]
```

FIGUUR 9 DNS QUERY NOS.nl

Het bijbehorende antwoord is weergegeven in Figuur 10. Het antwoord geeft één CName (alias) terug en vier IP-adressen. Deze voorbeelden komen uit de voor mijn onderzoek gebruikte dataset.

FIGUUR 10 DNS RESPONSE NOS.nl

```
Domain Name System (response)
    Transaction ID: 0x8485
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 5
    Authority RRs: 0
    Additional RRs: 0
  > Queries
  v Answers
    v www.nos.nl: type CNAME, class IN, cname d1wkry09pahjt9.cloudfront.net
          Name: www.nos.nl
          Type: CNAME (Canonical NAME for an alias) (5)
          Class: IN (0x0001)
          Time to live: 55 (55 seconds)
          Data length: 31
          CNAME: d1wkry09pahjt9.cloudfront.net
    v d1wkry09pahjt9.cloudfront.net: type A, class IN, addr 65.9.73.28
          Name: d1wkry09pahjt9.cloudfront.net
          Type: A (Host Address) (1)
          Class: IN (0x0001)
          Time to live: 59 (59 seconds)
          Data length: 4
          Address: d1wkry09pahjt9.cloudfront.net (65.9.73.28)
    v d1wkry09pahjt9.cloudfront.net: type A, class IN, addr 65.9.73.50
          Name: d1wkry09pahjt9.cloudfront.net
          Type: A (Host Address) (1)
          Class: IN (0x0001)
          Time to live: 59 (59 seconds)
          Data length: 4
          Address: d1wkry09pahjt9.cloudfront.net (65.9.73.50)
    v d1wkry09pahjt9.cloudfront.net: type A, class IN, addr 65.9.73.14
          Name: d1wkry09pahjt9.cloudfront.net
          Type: A (Host Address) (1)
          Class: IN (0x0001)
          Time to live: 59 (59 seconds)
          Data length: 4
          Address: d1wkry09pahjt9.cloudfront.net (65.9.73.14)
    v d1wkry09pahjt9.cloudfront.net: type A, class IN, addr 65.9.73.13
          Name: d1wkry09pahjt9.cloudfront.net
          Type: A (Host Address) (1)
          Class: IN (0x0001)
          Time to live: 59 (59 seconds)
          Data length: 4
          Address: d1wkry09pahjt9.cloudfront.net (65.9.73.13)
    [Request In: 52]
```

3.5 Kenmerken van fast-flux en domain-flux

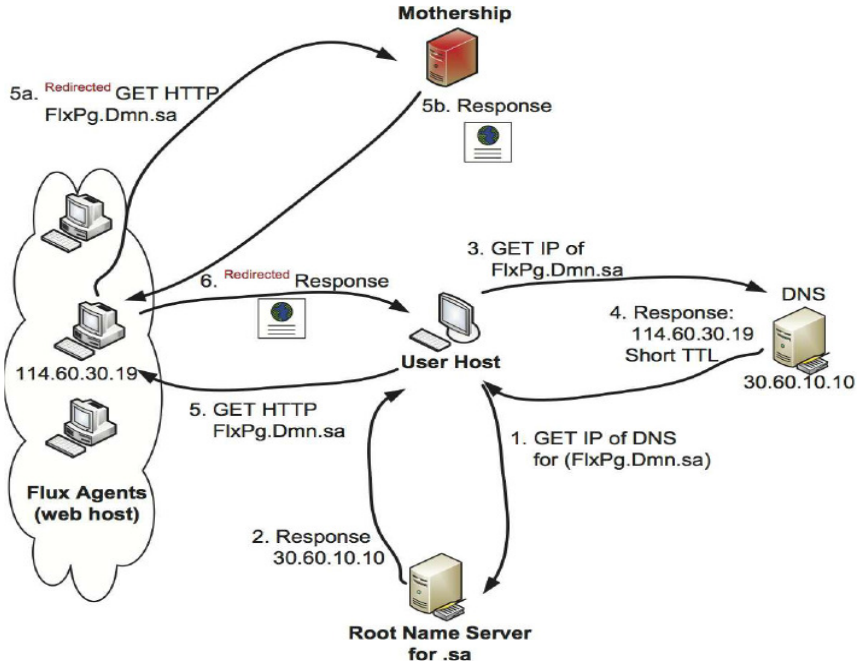
3.5.1 Fast-flux

Het basisconcept van fast-flux is de koppeling van een domeinnaam aan meerdere IP-adressen, waarbij de Time To Live (TTL) ingesteld is op een zeer korte periode. Wanneer geen gebruik gemaakt wordt van fast-flux ligt de TTL in het algemeen tussen de één en vijf dagen. Met deze periode heeft men profijt van DNS caching (Bilge, Kirda, Kruegel, & Balduzzi, 2011), (Bilge, Sen, Balzarotti, Kirda, & Kruegel, 2014). Bij fast-flux gebruik worden de IP-adressen continu in een hoog tempo gewisseld door middel van het aanpassen van DNS-records (Akamai, 2017), (Wang, Lin, Cheng, & Chen, 2017).
Overigens kan fast-flux ook ingezet worden voor legitieme doeleinden. Bijvoorbeeld voor 'loadbalancing' (het verdelen van de hoeveelheid verkeer over meerdere servers) door cloud-providers of systemen die een hoge beschikbaarheid moeten garanderen. Afhankelijk van de 'load' op een systeem zal het verkeer eventueel snel naar een ander systeem met minder belasting geleid worden. Een voorbeeld van loadbalancing zien we ook terug bij de website nos.nl (zie Figuur 10) waar het verkeer verdeeld wordt over 4 verschillende IP-adressen en daarmee over 4 verschillende DNS servers. Volgens Li et al. (Li, Wang, & Zhang, 2017) heeft Honeynet als een van de eersten waargenomen dat fast-flux technieken ingezet werden bij malafide praktijken zoals phishing, spam, malware etc.

Fast-flux kan worden onderverdeeld in single flux en double flux (Mahmoed, Nir, & Matrawy, 2014), (Hands, Yang, & Hansen, 2015), (Li, Wang, & Zhang, 2017):

- Single flux betekent dat alleen het IP-adres van een enkele domeinnaam in hoog tempo wordt veranderd. De IP-adressen hebben een zeer korte periode waarin ze geregistreerd staan.
- Double flux is een geoptimaliseerde versie van single flux. Hierbij worden niet alleen de IP-adressen van een domeinnaam in hoog tempo veranderd, maar ook de IP-adressen van de DNS-server (de NS-records) waar het domein is ondergebracht.

In Figuur 11 is schematisch weergegeven hoe een single flux DNS aanvraag eruit ziet (Mahmoed, Nir, & Matrawy, 2014, p. 6).



FIGUUR 11 SINGLE FLUX DNS AANVRAAG (M. MAHMOUD ET AL. 2014)

De stappen 1 tot en met 5 zijn hetzelfde als bij een normale DNS-aanvraag:

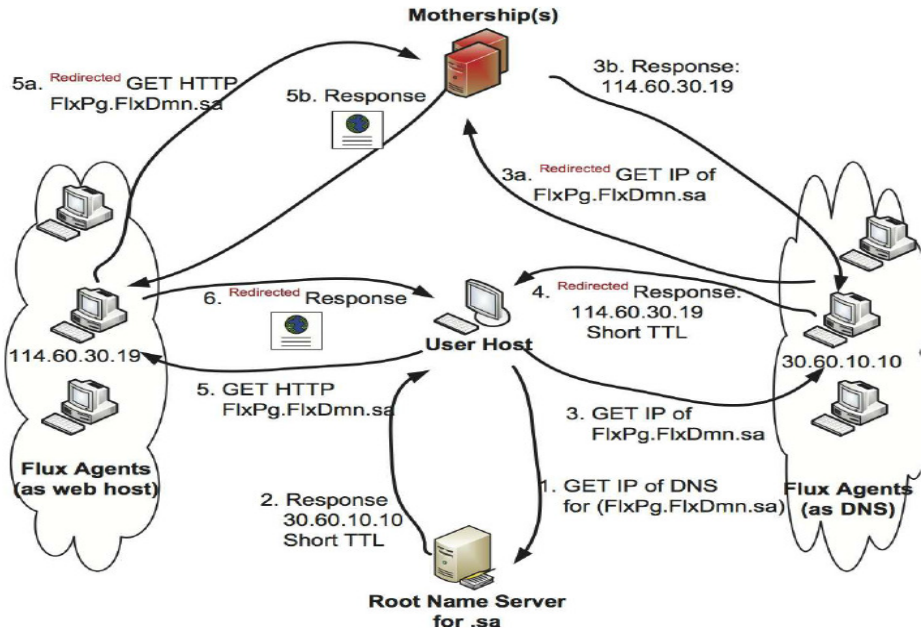
Stap 1: User host (gebruiker) vraagt bij de ".sa" root name server het IP-adres op van de DNS name server die "Dmn.sa" host.
Stap 2: In dit geval wordt het IP-adres 30.60.10.10 door de ".sa" root name server teruggegeven.
Stap 3: Vervolgens wordt door de user host het IP-adres 30.60.10.10 gebruikt om de betreffende DNS name server te contacten en het IP-adres voor "Pg.Dmn.sa" op te vragen.
Stap 4: De DNS name server geeft het IP-adres 114.60.30.19 retour. Het IP-adres heeft een zeer korte TTL waardoor bij een volgende opvraging het IP-adres weer opnieuw opgehaald moet worden en het IP-adres niet uit het cache geheugen van de DNS server gehaald kan worden.
Stap 5: De user host gebruikt het IP-adres 114.60.30.19 om de webserver te benaderen en informatie op te halen.

In stap 5 vinden twee tussenstappen plaats die worden uitgevoerd door 'flux agents', ook wel 'proxy bots' genoemd (Li, Wang, & Zhang, 2017), (Mahmoed, Nir, & Matrawy, 2014), (The Honeynet Project, 2008). Flux agents vormen een verzameling bots die de aanvraag van het IP-adres afhandelen alsof ze webserver zijn.

Stap 5a: De verzameling flux agents stuurt de aanvraag door naar de echte webserver/ C&C-server (het "Mothership").
Stap 5b: Die stuurt het antwoord terug naar de proxy bots.
Stap 6: Op hun beurt sturen deze dit antwoord door naar de user host.

De botmaster wisselt de verzameling flux agents continu. Dit wisselen van flux agents waarbij elke nieuwe aanvraag weer naar een andere agent wordt doorverwezen, wordt wel 'round robin' genoemd. Het wisselen van de IP-adressen gebeurt door middel van het aanpassen van het DNS-A-record op de Authoritive Name Server (IP-adres 30.60.10.10 in Figuur 11 en Figuur 12Figuur 2). Deze server wordt beheerd door een botmaster of door een andere partij waarbij de botmaster in staat is om de configuratie aan te passen. De server wordt gehost in een 'bullet proof' omgeving. Dit zijn hosting partijen die geen strenge eisen stellen aan het soort activiteiten dat plaats vindt op hun omgeving. In de DNS-logfiles is te zien dat IP-adressen met grote regelmaat wisselen in het DNS-A-record. Het blokkeren van een malafide IP-adres wordt hiermee zeer lastig.

In Figuur 12 hieronder, is schematisch weergegeven hoe een double flux DNS aanvraag eruitziet (Mahmoed, Nir, & Matrawy, 2014, p. 7).



FIGUUR 12 DOUBLE FLUX DNS REQUEST (M. MAHMOUD ET AL. 2014)

Bij double flux wordt het IP-adres van de malafide webserver voortdurend gewisseld en worden flux agents ingezet. Aanvullend wordt het IP-adres van de Authoritive Name server, welke wordt beheerd door de botmaster, ook continue gewisseld. In Figuur 12 is dit terug te zien in de toegevoegde stappen 3a en 3b.

Stap 1: de user host vraagt aan de '.sa root name server' het IP-adres van FlxPg.Dmn.sa.
Stap 2: een IP-adres met een zeer korte TTL wordt retour gegeven van de DNS-server die het domein Dmn.sa beheert. Door de korte TTL zal elke aanvraag een ander IP-adres opleveren.
Stap 3: wanneer de user host de DNS-server benadert, wordt in werkelijkheid een verzameling van flux agents/ proxy bots benaderd.
Stap 3a: deze verzameling vraagt bij de C&C-server (mothership) het IP-adres op (stap 3a).
Stap 3b: de C&C-server stuurt een IP-adres retour, in dit geval 114.60.30.19.
Stap 4: de verzameling van flux agents stuurt het antwoord door naar de user host.

De stappen 5a, 5b en 6 zijn vervolgens hetzelfde als in het single-flux schema.

Omdat elke aanvraag leidt tot een ander IP-adres, zijn Command & Control servers zeer lastig te ontdekken. Daarnaast heeft de botmaster de mogelijkheid een C&C-server zeer snel te verplaatsen, door middel van het aanpassen van (een) IP-adres(sen) zonder dat dit leidt tot verminderde beschikbaarheid.

Op basis van de opgedane kennis verwacht ik in de DNS logging de volgende karakteristieken te zien wanneer gebruik wordt gemaakt van fast-flux:

- ANCount > 2. Zowel bij legitiem gebruik als bij botnets die fast-flux gebruiken.
- Een korte TTL in het Answer, Authority en Additional gedeelte. Dit is te zien zowel bij legitiem gebruik als bij malafide gebruik.
- Meerdere IP-adressen in de secties Answer en Additional. De IP-adressen zullen een volgende keer wanneer het domein weer wordt opgevraagd, (deels) veranderd zijn. Bij legitiem gebruik zullen de IP-adressen alleen veranderd worden wanneer de load naar een andere machine moet worden gebracht. Bovendien zal het aantal wijzigingen in IP-adressen voor een DNS-server waarschijnlijk veel lager zijn.
- Aantal ASN behorend bij de IP-adressen. Bij legitiem gebruik zullen de verschillende IP-adressen over het algemeen behoren bij één Autonomous System Number (ASN). Elk AS heeft een eigen nummer: het Autonomous System Number. Een ASN is een groep IP-netwerken die beheerd wordt door één partij, ook wel een routeringsdomein genoemd. IP-adressen gekoppeld aan een malafide domein zullen mogelijk verspreid zijn over diverse ASN's. Dit houdt in dat een dergelijk domein bereikbaar is via verschillende IP-netwerken, oftewel providers.
- Geografische spreiding: bij legitiem gebruik zal de geografische spreiding relatief beperkt zijn. Zo zullen grote cloud-providers gebruikers uit bijvoorbeeld Nederland leiden naar een domein met een IP-adres uit de regio Europa.

3.5.2 Domain-flux

Domain-flux is het tegenovergestelde van fast-flux: veel domeinnamen bij één IP-adres versus één domeinnaam bij meerdere IP-adressen. Het gebruik van domain-flux wordt vanaf 2007 steeds meer gezien. Voorbeelden hiervan zijn botnets met namen als Conficker.A/B/C, Srizbi, Mjuyh, Kraken, Murofet en Torpig (Truong & Cheng, 2016), (Wang, Lin, Cheng, & Chen, 2017). Bij domain-flux wordt gebruik gemaakt van een Domain Generation Algorithm (DGA),waarmee geautomatiseerd een stroom van domeinnamen kan worden gegenereerd. Een deel van deze domeinnamen wordt tijdelijk geregistreerd door de botmaster om ingezet te worden voor het

continue verplaatsen van de Command & Control server. Bots proberen vervolgens aan de hand van de gegenereerde lijst met domeinnamen de C&C-server te bereiken, totdat ze een antwoord krijgen.
Het aantal domeinnamen dat gegenereerd wordt, kan aanzienlijk zijn. Zo is bekend dat botnet Conficker.C per dag 50.000 verschillende domeinnamen genereert, verdeeld over 110 top-level domeinen waarbij slechts enkelen door de botmaster geregistreerd zijn (.nl, .com etc.) (Li, Wang, & Zhang, 2017), (Sharifnya & Abadi, 2015), (Stone-Gross, et al., 9-13 november 2009). Hierdoor is het zeer lastig om een C&C-server te vinden en, als hij gevonden is, te ontmantelen of te blokkeren. Tegen de tijd dat een actief malafide domein is achterhaald, is de C&C-server naar alle waarschijnlijkheid alweer verplaatst.
Tegelijkertijd laat een botnet onmiskenbaar sporen achter in de DNS-logfiles, namelijk de vele DNS aanvragen. Daarnaast zijn veel van de gegenereerde domeinnamen niet geregistreerd en zullen een NXDomain (niet bestaand domeinnaam) als resultaat geven (Sharifnya & Abadi, 2015).

Voor het automatisch genereren van domeinnamen zijn vele variaties mogelijk. Wel moet worden voorkomen dat dubbele namen worden geregistreerd of dat reeds bestaande namen opnieuw worden geregistreerd. Bijvoorbeeld, het Botnet Conficker.C maakt gebruik van de huidige datum en tijd als domeinnaam (Sharifnya & Abadi, 2015), (Wang, Lin, Cheng, & Chen, 2017). Het Kraken botnet maakt gebruik van Engelse woorden of woorden die hierop lijken, gecombineerd met een willekeurig achtervoegsel zoals bijvoorbeeld –able, -dom, -ship, -ly, etc. (Sharifnya & Abadi, 2015), (Wang, Lin, Cheng, & Chen, 2017).

Er zijn vele varianten van domain-flux. Belangrijke kenmerken hierbij zijn:

1. Grote hoeveelheden domeinnamen behorend bij eenzelfde IP-adres (Bilge, Kirda, Kruegel, & Balduzzi, 2011), (Hands, Yang, & Hansen, 2015).
2. Bij een deel van de domain-flux botnets verschillen de automatisch gegenereerde domeinnamen significant van domeinnamen die door mensen zijn bedacht. Dit geldt bijvoorbeeld niet voor domain-flux botnets die gebruik maken van zogeheten wordlist-based DGA's. Bij deze worden goed leesbare domeinnamen geconstrueerd (Sharifnya & Abadi, 2015).

In tegenstelling tot fast-flux technieken worden domain-flux technieken niet voor legitieme doeleinden gebruikt. Op basis van de werking van domain-flux, verwacht ik dat wanneer in een ISP netwerk domain-flux wordt toegepast door botnets, in de DNS logging het volgende is te zien:

- Grote hoeveelheid NXdomains-antwoorden in een bepaalde tijdsperiode, dat wil zeggen opgevraagde domeinnamen die niet bestaan. Deze antwoorden worden veroorzaakt doordat een bot een hele lijst doorwerkt voordat hij bij een domeinnaam komt die wel geregistreerd is door de botmaster.
- Taal-technische verschillen tussen door DGA gegenereerde domeinnamen en door mensen bedachte domeinnamen.
- Recente leeftijd van een domeinnaam. Omdat door DGA gegenereerde domeinnamen maar kort geregistreerd zijn, zal het tijdstip waarop het domein is aangemaakt altijd recent zijn.
- Lage TTL: een domein is maar kortstondig geregistreerd.

3.6 Conclusie

Botnets zijn door de jaren heen geëvolueerd tot de ruggengraat van de cybercriminaliteit. Ze zijn qua infrastructuur sterk ontwikkeld en daardoor zeer lastig te detecteren en te ontmantelen. Wanneer botnets zowel fast-flux als domain-flux gebruiken worden de mogelijkheden om een C&C server uit de lucht te halen tot een minimum beperkt.

Fast-flux en domain-flux hebben geen overeenkomende kenmerken. Het zijn technieken waarbij de ene uit gaat van het wisselen van de IP-adressen (fast-flux) en de andere gericht is op het

wisselen van de domeinnaam (domain-flux). Fast-flux kan ook voor legitieme doeleinden ingezet worden, bijvoorbeeld voor het loadbalancen van internetverkeer. Domain-flux wordt niet gebruikt voor legitieme doeleinden.

Praktisch elk botnet maakt tegenwoordig gebruik van DNS. Uit het literatuuronderzoek wordt duidelijk dat de huidige generatie botnets vooral beide technieken hanteren (Singh, Singh, & Kaur, 2019) (Zhauniarovich, Khalil, Yu, & Dacier, 2018). Dit gebruik van beide technieken geeft een bepaald patroon in de DNS aanvragen. Dit patroon is anders dan bij legitieme aanvragen. Door een combinatie te nemen van de kenmerken TTL, ouderdom van een domeinnaam, verschillende AS nummers, verschillende ASN registrar en herkomst van IP-adressen, zijn deze botnets te herkennen. In mijn onderzoek maak ik gebruik van deze kenmerken.

4 MACHINE LEARNING EN BOTNETDETECTIE

Machine learning is de term voor een geheel van meerdere leertechnieken die het laatste decennium steeds meer in opkomst is. Ook bij botnetdetectie wordt deze techniek steeds meer toegepast. In deze paragraaf wordt het ontstaan van machine learning beschreven en een overzicht gegeven van hoe het gebied is ingedeeld. Dit wordt gevolgd door een bespreking van literatuur over de inzet van machine learning bij de detectie van botnets gebaseerd op DNS data. Tot slot zal ik toelichten hoe machine learning is ingezet bij mijn onderzoek.

4.1 Achtergrond van Machine Learning

Kunstmatige intelligentie is een overkoepelende term voor systemen of machines die de menselijke intelligentie nabootsen. Kunstmatige intelligentie en machine learning (ML) worden vaak in één adem genoemd alsof het twee woorden voor hetzelfde onderwerp zijn. Maar ML is een onderdeel van kunstmatige intelligentie en kunstmatige intelligentie omvat meer.
Veel mensen denken dat het bij machine learning gaat over geheel zelfstandig lerende en opererende systemen. Echter, het systeem leert altijd op basis van door mensen aangeboden data. Het zijn dan ook mensen die de kaders aangeven en de wijze waarop geleerd wordt. In hoofdstuk 5 zal nader worden in gegaan op deze data.
Bij supervised machine learning, onderdeel van machine learning, wordt gewerkt worden met gelabelde data. Labelen van data is het toevoegen van zinvolle informatie om context te geven aan het ML systeem. In de ideale situatie wordt dit uitgevoerd door deskundigen op het bestudeerde onderwerp. Hiermee zijn ook weer mensen degenen die uiteindelijk bepalen hoe een ML systeem leert. Er is dus eigenlijk geen sprake van machine learning, van automatisch leren, maar van een learning machine: een lerende machine waarbij mensen steeds grote invloed hebben op wat en hoe de machine leert.
Op basis van definities die door Samuel (Samuel, 1959) en Mitchel (Mitchel, 1997) zijn opgesteld kan machine learning als volgt samengevat worden:

Machine learning is een toepassingsgebied binnen kunstmatige intelligentie dat gebruik maakt van statistische technieken waarmee computersystemen zichzelf verbeteren op basis van aan-geboden data en opgedane ervaring hiermee, zonder dat deze expliciet geprogrammeerd wordt.

In de literatuur wordt Alan Turing gezien als een van de grondleggers van kunstmatige intelligentie en Arthur Samuel als de grondlegger van machine learning. De rol van Ada Lovelace wordt hierbij in de literatuur tekort gedaan.
Ada Lovelace (1815 – 1852) was een Britse wiskundige, gefascineerd door de ideeën van Charles Babbage (1791 – 1871) over een zogenaamde 'Analytical Engine', veel later werd dit ook wel de eerste mechanische computer genoemd. Babbage was vooral geïnteresseerd in de rekenkundige mogelijkheden van dit apparaat. Lovelace zag destijds al in dat een dergelijke machine tot meer in staat zou zijn. Ze vertaalde een samenvatting van Babbages plannen uit het Frans naar het Engels en breidde deze samenvatting uit met haar eigen aantekeningen. Deze uitbreiding omvatte onder andere het tot in detail beschrijven hoe de analytische machine met behulp van een algoritme Bernouille getallen (veel voorkomende wiskundige getallenreeks in de getaltheorie) kon berekenen. Deze beschrijving wordt nu gezien als het eerste computerprogramma. Immers, een computerprogramma kan worden gezien als de implementatie van een beschreven algoritme. De vertaling en haar uitgebreide aanvullingen publiceerde Lovelace als artikel in 1843. Zij voorzag dat een dergelijke analytische machine ook gebruikt zou kunnen worden om muziek te componeren, afbeeldingen te maken en wetenschap te bedrijven (Wikipedia, 2021). Hoewel Lovelace zeer hoge verwachtingen had van een dergelijke machine, was creatief denken daar niet een van. Ze was van mening dat een 'analytical engine' niet zelf iets kan voortbrengen. Het kan alleen uitvoeren wat mensen weten en hoe het uitgevoerd moet worden: onze kennis en uitvoeringsinstructies. Ze geeft

zelfs een waarschuwing ervoor te waken dat geen overdreven ideeën ontstaan over de mogelijkheden van de Analytical Engine. (Mindmatters, 2020).

Alan Turing was van mening dat computers wel degelijk konden denken. In 1950 publiceerde hij een artikel waarin de eerste zin was: "I propose to consider the question, 'Can machines think?' " (Turing, 1950). Hiermee weerlegde hij de bewering van Lovelace dat computers niet in staat zouden zijn zelfstandig te denken. Het artikel beschrijft de Turing-test die gebaseerd is op het idee dat als mens-machine communicatie niet kan worden onderscheiden van mens-mens communicatie, dit het bewijs is dat een machine zelfstandig in staat is om te leren. Volgens Turing blijkt hieruit dat een computer wel in staat is om zelf te denken en tot nieuwe conclusies te komen die mensen nog niet hadden voorzien (Turing, 1950), (Mindmatters, 2020).
Mijns inziens, komt Ada Lovelace's standpunt meer overeen met wat de huidige mogelijkheden zijn van machine learning dan het standpunt van Turing. Een systeem leert op basis wat mensen aan het systeem aanbieden (de data), op een door hen gekozen manier (het gekozen algoritme).
De Turing-test heeft geleid tot vele nieuwe ideeën en systemen. Arthur Samuel (IBM) ontwikkelde rond 1952 het eerste machine learning algoritme voor een schaakprogramma (Samuel, 1959). Een algoritme is een wiskundig recept om vanuit een gegeven begintoestand naar een beoogd doel te komen op basis van een eindige reeks van instructies. Algoritmen fungeren dan ook als de motor voor machine learning. Frank Rosenblatt ontwikkelde vervolgens in 1958 het eerste perceptron-algoritme (Rosenblatt, 1958). Een perceptron is een neuraal netwerk, met één invoerlaag, één of meer verborgen lagen en één uitvoerlaag. Het Rosenblatt perceptron bestaat uit één verborgen laag. Dit algoritme is nog steeds de basis van alle neurale netwerken.

In het afgelopen decennium heeft machine Learning een enorme groei doorgemaakt. Het is tegenwoordig niet meer weg te denken in onze samenleving. We maken dagelijks gebruik van deze technieken zonder dat we dit in de gaten hebben, bijvoorbeeld door het gebruik van Facebook, SIRI spraakherkenning, spamdetectie, aanbevelingen voor Netflix, etc. Ook op andere gebieden, zoals in wetenschap, technologie, commercie of binnen de zorg wordt ML veel toegepast. Volgens Jordan en Mitchell heeft deze enorme groei te maken met een drietal ontwikkelingen (Jordan & Mitchell, 2015):

1) 'Big Data': op dit moment zijn meer gegevens dan ooit beschikbaar. Deze zogenaamde Big Data vraagt dan ook specifieke technieken om hier goed mee om te kunnen gaan.

2) 'Big Compute': computers zijn vele malen krachtiger geworden en daarnaast zijn de kosten van rekenkracht de afgelopen jaren sterk gedaald. De kosten voor computers halveren elk jaar en de rekenkracht verdubbelt elk jaar.

3) Algoritmen: de algoritmen die gebruikt worden voor machine learning zijn sterk verbeterd. Mede door de toegenomen rekenkracht van computers zijn de mogelijkheden om deze techniek toe te passen sterk vergroot.

In de literatuur worden de termen machine learning algoritme en machine learning model door elkaar gebruikt. Het verschil is ook lang niet altijd duidelijk. In deze scriptie bedoel ik met machine learning algoritme een recept voor het oplossen van een probleem. Het machine learning model is het geheel van dataset, algoritme en bijbehorende parameters die van belang zijn bij een gekozen algoritme.

4.2 Indeling van Machine Learning Algoritmen

Een mogelijke indeling voor machine learning algoritmen is de wijze waarop een algoritme wordt getraind (leerparadigma's) waarbij vier hoofdvormen kunnen worden onderscheiden (Géron, 2017), (Stevanovic & Pedersen, 2013), (Hoang & Nguyen, 2018):

Supervised learning
Bij supervised learning bevat de aangeboden dataset de gegevens voor een gewenste oplossing: dit wordt een gelabelde dataset genoemd. Het machine learning algoritme wordt geschoold door middel van een trainingsproces waarbij gecorrigeerd wordt als de voorspellingen niet kloppen. De training stopt als de gewenste nauwkeurigheid is bereikt. Supervised learning wordt veel toegepast bij classificatie problemen, bijvoorbeeld het classificeren van e-mail als spam of niet. Ook het detecteren van botnets kan als een classificatie probleem omschreven worden.

Het kiezen van de juiste kenmerken (features) is een van de belangrijkste onderdelen van het model. Wanneer niet goed gekozen features worden gebruikt werkt het model niet naar behoren. Bijvoorbeeld wanneer de verkeerde features worden gekozen voor het herkennen van een auto: niet een auto wordt herkend, maar elk voertuig dat rijdt op de weg.

Unsupervised learning
Bij unsupervised learning is de dataset niet gelabeld, dat wil zeggen dat het resultaat niet vooraf bekend is. Dit betekent dat het model leert zonder trainer (labels) en zonder dat vooraf duidelijk is wat het resultaat precies zal zijn. Het model wordt geschoold om in de dataset aanwezige structuren te ontdekken en herkennen. Unsupervised learning wordt vaak toegepast bij clustering, bijvoorbeeld het groeperen van soorten bezoekers op een symposium. Vooraf wordt niet aangegeven tot welke groep een bezoeker zou moeten horen. Het model zoekt dit zelf uit.

Semi-supervised learning
Bij semi-supervised learning is de dataset deels gelabeld en deels niet gelabeld. Over het algemeen zijn semi-supervised modellen een combinatie van supervised algoritmen en unsupervised algoritmen. Bijvoorbeeld, bij het detecteren van botnets wordt het probleem in tweeën opgedeeld. Perdisci et al. groeperen eerst domeinnamen met behulp van clustering (unsupervised) om vervolgens deze domeinnamen te laten classificeren (supervised) als malafide of bonafide (Perdisci, Corona, Dagon, & Lee, 2009).

Reinforcement learning
Reinforcement learning is een geheel andere vorm, namelijk één waarbij het model leert van zijn omgeving. Hij wordt getraind om zijn omgeving te observeren, acties te selecteren en uitvoeren en krijgt dan een beloning of straf. Zo leert het model wat de beste strategie is.

Voor een goede werking van machine learning is het van belang dat het trainingsmateriaal een correcte afspiegeling is van de werkelijkheid. De training dataset zal een realistische weergave moeten zijn van datgene wat onderzocht dient te worden.

Binnen het veld botnetdetectie wordt zowel gebruik gemaakt van supervised, unsupervised als semi-supervised learning. Reinforcement learning wordt tot nu toe weinig gebruikt. (Alieyan, Almomani, Manasrah, & Kadhum, 2017), (Acarali, Rajarajan, Komninos, & Herwono, 2016), (Garcia, Zunino, & Campo, 2014), (Hoang & Nguyen, 2018).

4.3 Deep Learning

Deep learning (DL) is een speciaal onderdeel van machine learning en wordt gekenmerkt door het gebruik van verschillende neurale lagen, geïnspireerd op de biologische neuronen: input layer (invoer), hidden layer (kan uit 1 of meerdere lagen bestaan), output layer (uitvoer).

Tussen deep learning en machine learning zijn volgens Géron de volgende verschillen:

- DL algoritmen nemen meer rekenkracht dan ML algoritmen vanwege de vele matrix berekeningen die nodig zijn. Hoe meer lagen een netwerk heeft, hoe meer rekenkracht er nodig is.
- Over het algemeen hebben deep learning algoritmen meer trainingstijd nodig.
- Wanneer datasets zeer groot zijn, zijn deep learning algoritmen sneller dan andere machine learning algoritmen. Het selecteren van de juiste features, vaak ook een tijdrovende bezigheid, is bij DL algoritmen sneller, terwijl er ook minder features nodig zijn.
- Wanneer de dataset zeer groot is, werken deep learning algoritmen over het algemeen beter dan machine learning algoritmen (Géron, 2017).

Waarbij opgemerkt moet worden dat de keus voor DL of ML ook afhangt van beschikbare middelen, tijd, dataset en de taak die moet worden volbracht.

Deep learning wordt veel toegepast bij spraak- en beeldherkenning en bij tekstanalyse zoals bijvoorbeeld bij sociale media (Van Roosmalen, 2017).
Bij botnetdetectie wordt vooral gebruik gemaakt van machine learning (Ring M. , Wunderlich, Scheuring, Landes, & Hotho, 2019), (Alieyan, Almomani, Manasrah, & Kadhum, 2017). In Tabel 1, paragraaf 4.4, geef ik een overzicht van ML algoritmen die gebruikt worden bij DNS botnetdetectie. Het merendeel van de onderzochte studies maakt gebruik van machine learning in plaats van deep learning. Het onderzoek van Xu et al. is een van de weinige voorbeelden waar gebruik gemaakt wordt van deep learning (Xu, Shen, & Du, 2019). Buiten het veld van DNS botnetdetectie constateert Van Roosmalen dat hij een van de eersten is geweest die DL voor P2P botnetdetectie heeft toegepast (Van Roosmalen, 2017).

4.4 Type probleem en bijbehorend algoritme

In paragraaf 4.2 is toegelicht op welke wijze een machine kan leren (leerparadigma). Er zijn voorbeelden genoemd van algoritmen die geschikt zijn voor de verschillende manieren van leren. Het juiste machine learning algoritme wordt niet alleen op basis van de manier van leren gekozen. Ook het type probleem dat opgelost moet worden, is van belang bij de keuze van het algoritme, want algoritmen zijn niet per definitie passend voor elke situatie.
Algoritmen kunnen gegroepeerd worden in drie verschillende groepen:

- Regressie algoritmen: deze zijn geschikt voor vraagstukken waarbij numerieke waarden voorspeld moeten worden aan de hand van diverse inputvariabelen. Bijvoorbeeld, hoe lang een elektrische auto gemiddeld kan rijden onder bepaalde omstandigheden.
- Classificatie algoritmen: zijn geschikt voor vraagstukken waarbij voorspeld moet worden in welke klasse een element hoort. Wanneer er maar twee klassen zijn, dan wordt dit een binair classificatie model genoemd.
- Clustering algoritmen: worden gebruikt voor het ontdekken van structuren of afwijkingen in data en voor het groeperen van objecten in clusters (Scikit-learn, 2011), (Géron, 2017).

Binnen het vakgebied botnetdetectie worden met name classificatie en clustering algoritmen toegepast. De classificatie algoritmen worden vooral gebruikt in situaties waar malafide domeinnamen gedetecteerd moeten worden. Clustering algoritmen worden vooral toegepast wanneer gezocht wordt naar afwijkingen in dataverkeer.

In Tabel 1 geef ik een overzicht van onderzoeken naar DNS botnetdetectie met behulp van machine learning en de toepaste machine learning algoritmes. Hierin is tevens aangegeven van welke leerparadigma gebruik is gemaakt.

TABEL 1 OVERZICHT DNS BOTNETDETECTIE MET BEHULP VAN MACHINE LEARNING

Detectie Methode	*Jaar*	*Supervised / Unsupervised / Hybride*	*Gebruikte MLA's*
FluXOR (Passerini, Paleari, Martignoni, & Bruschi, 2008)	2008	Supervised	naïve Bayesian classifier
Fast-flux Monitor (Caglayan, Toothaker, Drapeau, Burke, & Eaton, 2009)	2009	Supervised	Bayesian belief networks
Perdisci et al. (Perdisci, Corona, Dagon, & Lee, 2009)	2009	Hybride	C4.5 decision tree classifier, hierarchical clustering algoritm
Notos (Antonakakis, Perdisci, Dagon, Lee, & Feamster, 2010)	2010	Hybride	Network based clustering (network based vectors), zone based clustering (zone based vectors), cluster characterization (kNN)
Exposure (Bilge, Kirda, Kruegel, & Balduzzi, 2011)	2011	Supervised	decision tree
Kopis (Antonakakis, Perdisci, Lee, Vasiloglou, & Dagon, 2011)	2011	Supervised	decision tree
Digger (Hu, Knysz, & Shin, 2011)	2011	Supervised	Support Vector Classifier
BotGAD (Choi & Lee, 2012)	2011	Unsupervised	X-means clustering
Pleiades (Antonakakis, et al., 2012)	2012	Hybride	X-means clustering en decision tree
Mentor (Kheir, Tran, Caron, & Deschamps, 2014)	2014	Supervised	selective Bayesian classifier, SVM, J48, C4.5 decision trees
CROFlux (Grzinic, Perhoc, Maric, Vlasic, & Kulcsar, 2014)	2014	Supervised	Support Vector Machine
Phoenix (Schiavoni, Maggi, Cavallaro, & Zanero, 2014)	2014	Hybride	Spectral clustering algoritm, DBSCAN clustering algoritm.
DFBotKiller (Sharifnya & Abadi, 2015)	2015	Onbekend	J-S Bigram, SRCC Bigram, LD: een combinatie van Natural Language Processing en deep learning
Troung et al. (Truong & Cheng, 2016)	2016	Supervised	Meerdere getest: naïve Bayes, K-nearest neighbor, support vector machine, decision tree (J48), random forest
DBod (Wang, Lin, Cheng, & Chen, 2017)	2016	Unsupervised	Clustering: Chinese Whispers algorithm
Fast-flux Hunter (Almomani, 2018)	2016	Hybride	C4.5 decision tree, Evolving Fuzzy Neural Networks
Hoang en Nguyen (Hoang & Nguyen, 2018)	2018	Supervised	Meerdere getest: K-nearest neighbor, decision tree (C4.5), random forest
Xu et al. ((Xu, Shen, & Du, 2019)	2019	Supervised Deep Neural network.	N-gram in combinatie met deep convolutional neural network

4.5 DNS botnetdetectie en Machine Learning

In deze paragraaf wordt eerst een algemeen beeld geschetst van diverse botnetdetectie technieken voordat ingegaan wordt op detectiemethoden die specifiek op DNS gebaseerd zijn. Bij de DNS gebaseerde detectiemethoden wordt toegelicht in welke categorieën deze ingedeeld kunnen worden en welke kenmerken hierbij een rol spelen.

4.5.1 Overzicht botnetdetectie methoden

Het gebruik van machine learning bij botnetdetectie is gebaseerd op het feit dat botnets bepaalde datapatronen genereren die herkend kunnen worden. Er zijn verschillende indelingen voor botnetdetectie technieken. Een veel gebruikte indeling is die van Alieyan et al. (Alieyan, Almomani, Manasrah, & Kadhum, 2017) en Sing et al. (Singh, Singh, & Kaur, 2019). De technieken zijn onderverdeeld in twee hoofdcategorieën:

- Honeynet
- Intrusion Detection System technieken.

Bij honeynet wordt bewust een omgeving gebruikt die geïnfecteerd kan worden met (bekende) botnetvarianten zodat deze bestudeerd kunnen worden. Een honeynet is relatief eenvoudig op te

zetten en botnet data kunnen met zekerheid als kwaadaardig gelabeld worden. Een nadeel is dat er eigenlijk per botnet een eigen omgeving moet worden gecreëerd. Bovendien zijn botmasters steeds beter in staat om honeynets te detecteren en te omzeilen.

Intrusion Detection System technieken worden onderverdeeld in detectiemethoden gebaseerd op signatuur- en op anomalie (op afwijking). Bij op signatuur gebaseerde technieken wordt gekeken naar specifieke kenmerken van een botnet. Het botnet dient dan bekend te zijn. Onbekende botnets worden met deze methoden niet herkend evenals botnets die gebruik maken van encryptie technieken. Tegenwoordig maken de meeste botnets gebruik van encryptie technieken en veranderen botnets regelmatig hun signatuur (Singh, Singh, & Kaur, 2019), (Li, Wang, & Zhang, 2017).
In geval van op anomalie gebaseerde technieken wordt er gekeken naar afwijkingen in het netwerkverkeer. Wel moet in dit geval bekend zijn wat normaal verkeer is en wat afwijkend verkeer is. Het zoeken naar afwijkingen kan met behulp van specifieke apparaten (host gebaseerd) in het netwerk of op basis van netwerkverkeer. In het eerste geval worden individuele apparaten gemonitord op afwijkingen. De schaalbaarheid van host-gebaseerde detectie is dan ook beperkt en er worden alleen botnets gevonden die op het desbetreffende apparaat zitten.
In het tweede geval wordt er op een hoger niveau gekeken: op basis van afwijkingen in het netwerkverkeer kunnen botnets worden gedetecteerd. Deze detectie kan zowel actief als passief plaatsvinden.
Bij actieve detectie worden er pakketten in het netwerk geïnjecteerd waarop wordt geanalyseerd hoe hierop wordt gereageerd. Drie nadelen bij deze vorm van detectie:

1. Er wordt extra verkeer gegenereerd wat een extra belasting op het netwerk kan zijn.
2. Doordat er actief acties worden ondernomen, kunnen deze gedetecteerd worden door de botmaster waarna deze hierop kan reageren.
3. Met het actief manipuleren van data wordt grijs gebied betreden van wat toelaatbaar is. Met de nieuwe, strengere, privacy richtlijnen (GDPR) zijn de mogelijkheden om actief dataverkeer te monitoren aan strenge eisen onderworpen (Leenes, 2013), (Koops, 2013).

Met passieve detectiemethoden wordt er gedetecteerd op basis van kenmerken van het verkeer dat door het netwerk stroomt. Er wordt niet inhoudelijk naar de pakketten gekeken en dus niet ingegrepen, er wordt alleen gemonitord. De laatste jaren is er vooral veel aandacht voor dit gebied, mede door de nadelen van de andere gebieden. In Figuur 13 wordt de onderverdeling schematisch weergegeven (Silva, Silva, Pinto, & Salles, 2012). In het vakgebied botnetdetectie valt mijn onderzoek binnen het onderdeel DNS gebaseerde detectiemethoden. Dit is in Figuur 13 blauw gearceerd.

<table>
<tr><td colspan="7">Botnetdetectie Technieken</td></tr>
<tr><td>Honeynet gebaseerd</td><td colspan="6">Intrusion Detection System (IDS)</td></tr>
<tr><td rowspan="4"></td><td>Signatuur gebaseerd</td><td colspan="5">Anomalie gebaseerd</td></tr>
<tr><td rowspan="3"></td><td>Host gebaseerd</td><td colspan="4">Netwerk gebaseerd</td></tr>
<tr><td rowspan="2"></td><td>o.b.v. actieve monitoring</td><td colspan="3">o.b.v. passieve monitoring</td></tr>
<tr><td></td><td>IRC</td><td>DNS</td><td>SMTP</td><td>P2P</td><td>Divers</td></tr>
</table>

FIGUUR 13 OVERZICHT BOTNETDETECTIE TECHNIEKEN (SILVA, SILVA, PINTO, & SALLES, 2012)

4.5.2 DNS gebaseerde botnetdetectie methoden

Door de jaren heen zijn er verschillende vormen van op DNS gebaseerde botnetdetectie methoden ontwikkeld. Volgens Singh et al. (Singh, Singh, & Kaur, 2019) kunnen deze methoden verdeeld worden in 5 categorieën: Flow, Anomaly, Flux, DGA en Botinfectie.

Flow
Bij flow-gebaseerde technieken wordt geprobeerd om verkeersstromen te classificeren in kwaadaardig of goedaardig op basis van diverse parameters. Flow-gebaseerde technieken werken vooral goed voor het detecteren van botnets die gebruik maken van het IRC communicatie protocol.

Anomaly
Met op anomaly gebaseerde technieken kunnen afwijkingen gevonden worden in het verkeer of in de parameters van een netwerk. Een systeem dat geïnfecteerd is door een botnet zal afwijkend gedrag vertonen ten opzichte van een niet geïnfecteerd systeem. Bijvoorbeeld, een afwijking in de hoeveelheid DNS aanvragen of een lage TTL. Echter, doordat grote cloud-leveranciers loadbalancing toepassen en hiervoor een lage TTL gebruiken, kan deze parameter tot meer 'false positives' leiden.

Flux
Bij op Flux gebaseerde technieken wordt vooral gekeken naar de veranderingen in IP-adressen behorend bij een domeinnaam. Zoals eerder vermeld, gebruiken botnets fast-flux om IP-adressen snel te kunnen wisselen ter bescherming van het botnetwerk. De combinatie van parameter aantal IP-adressen en de parameter IP-geolocatie kan legitiem flux gebruik onderscheiden van malafide gebruik. De geografische spreiding van IP-adressen zal bij botnets groter zijn dan bij bonafide gebruik. Bij bonafide gebruik worden gebruikers geleid naar een server in eenzelfde regio voor een zo efficiënt mogelijke afhandeling van het verkeer. Bij malafide gebruik worden IP-adressen uit diverse regio's gebruikt om de C&C server zo goed mogelijk te verbergen.

Domain Generation Algoritme (DGA)
Bij op DGA gebaseerde detectie technieken wordt vooral gekeken naar karakteristieken van de opgevraagde domeinnamen. Hierbij wordt onder andere gekeken naar de opbouw van domeinnamen: de lengte en het gebruik van klinkers en medeklinkers. Bij deze detectietechniek kan gebruik gemaakt worden van 'whitelisting': domeinnamen die als vertrouwd kunnen worden beschouwd. 'Blacklisting', het uitsluiten van domeinnamen, is lastiger omdat domeinnamen heel snel weer worden opgeheven en blacklisting dan weinig zin meer heeft.

Botinfectie
Waar de meeste detectie technieken gericht zijn op een C&C infrastructuur, hebben botinfectie technieken als doel geïnfecteerde hosts te vinden binnen een netwerk. Het ontdekken van een C&C infrastructuur is vaak ontzettend lastig en tijdrovend, mede door het gebruik van DGA en fast-flux. Met het detecteren van geïnfecteerde hosts kan gelijk actie ondernomen worden en hoeft niet meer gewacht te worden totdat een C&C server wordt ontmanteld. Bij botinfectie technieken wordt ook gebruik gemaakt van whitelisting en blacklisting Volgens Singh et al. zijn botinfectie detectie technieken nog vrij jong zijn. Er zijn nog weinig gelabelde datasets beschikbaar voor deze techniek en ook het vaststellen van de labels (ground truth) is een grote uitdaging (Singh, Singh, & Kaur, 2019).

Bovenstaande technieken zijn in Tabel 2 weergegeven, inclusief de parameters die hiervoor gebruikt kunnen worden.

TABEL 2 DNS GEBASEERDE BOTNETDETECTIE METHODEN

DNS gebaseerde botnetdetectie methoden				
Flow	Anomaly	Flux	DGA	Botinfectie
Parameters (geen uitputtende opsomming):				
Aantal pakketten	Lage TTL waarde	Lage TTL waarde	Aantal Top Level domains	Whitelist domeinnamen / IP-adressen
Totaal aantal pakketten	Aantal mislukte DNS queries	Aantal IP-adressen	Aantal second Level domains	Blacklist domeinnamen / IP-adressen
Bytes / pakket	Flexibele DNS-IP toewijzingen	Aantal wisselende IP-adressen	Aantal NX-answers	IP-adressen
Bytes / seconde		Aantal ASN's	Aantal domeinen behorend bij IP-adres	FQDNS
Aantal drieweg TCP handshakes		Geografische spreiding IP-adressen	Ouderdom domeinnaam	DNS Mapping
Aantal verbroken verbindingen		Aantal queries	Taalaspecten domeinnaam	
			Lengte domeinnaam	

In mijn onderzoek maak ik gebruik van een combinatie van deze 5 categorieën: zowel kenmerken van Flux als van DGA, omdat beide technieken worden gebruikt om botnets te beschermen. Het doel van mijn onderzoek is om fast-flux en domein-flux botnets te detecteren. Daarnaast kan met deze detectie ook botnetinfecties worden vastgesteld. Wanneer een domeinnaam als malafide is aangemerkt, kunnen alle IP-adressen van hosts die een dergelijke domeinnaam opvragen als bot worden aangemerkt.

4.6 Machine Learning en mijn onderzoek

Om fast-flux en domain-flux botnets te detecteren, moet het door mij op te zetten model opgevraagde domeinnamen kunnen beoordelen op kenmerken van fast-flux en domain-flux technieken. Als een domeinnaam voldoet aan deze kenmerken dan wordt het domein aangemerkt als malafide. Dit is een voorbeeld van een binair classificatie probleem: een domein is bonafide of malafide. Op grond van overwegingen van tijd en rekenkracht heb ik gekozen voor een relatief beperkte dataset. Hiervoor zijn machine learning algoritmen een betere keus dan deep learning algoritmen.

4.6.1 Algoritme selectie

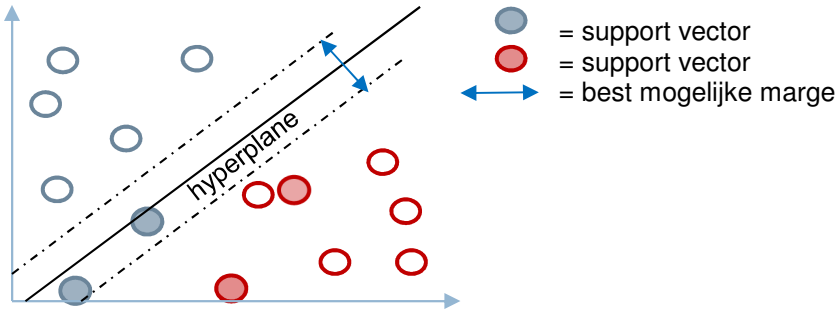
Wanneer machine learning wordt toegepast, wordt een aantal verschillende algoritmen, passend bij de vraagstelling geselecteerd om zo tot een goede keus te komen. In mijn onderzoek beperk ik mij tot de meest toegepaste classificatie algoritmen die hieronder kort beschreven worden:

- Support Vector Machine (SVM)
- Decision Trees
- Random Forests
- Naive Bayes
- K-Nearest Neighbors (k-NN)
- Logistic Regression (Géron, 2017), (Hoang & Nguyen, 2018), (Pedregosa, et al., 2011).

Hieronder worden de algoritmen nader toegelicht. Deze toelichting is gebaseerd op Scikit en Géron (Scikit-learn, 2011), (Géron, 2017).

4.6.1.1 SUPPORT VECTOR MACHINE

Een Support Vector Machine (SVM) is een krachtig en veelzijdig algoritme dat voor zowel classificatie- als regressie-vraagstukken gebruikt kan worden. Dit algoritme wordt ook ingezet voor het detecteren van afwijkingen. Een SVM verdeelt data in groepen met behulp van zogenaamde scheidingsvlakken, hyperplanes genoemd. De hyperplanes worden gecreëerd door gebruik te maken van support vectors. Support vectors zijn de datapunten die het dichtst bij het scheidingsvlak liggen. De afstand tussen de support vectors en de hyperplane wordt de marge genoemd. De hyperplane scheidt de verschillende groepen data van elkaar. In Figuur 14 is dit schematisch weergegeven.



FIGUUR 14 SCHEMATISCHE WEERGAVE VAN EEN SUPPORT VECTOR MACHINE ALGORITME

Voordelen:

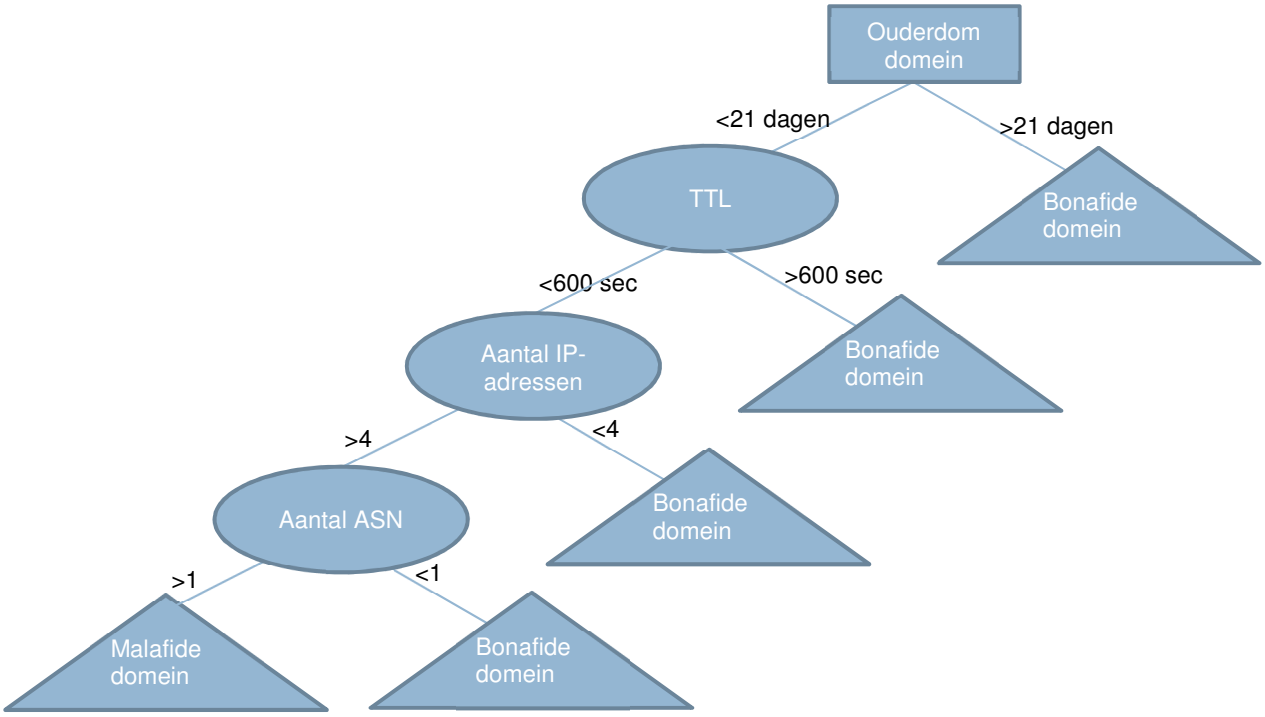
- Is zeer nauwkeurig bij schone datasets.
- Werkt snel bij kleinere datasets.
- Weinig risico op 'overfitting', het aanpassen aan de trainingsdata dat een model nieuwe data slecht kan classificeren.
- Werkt zeer goed bij tekstclassificatie.

Nadelen:

- 'Black box': specifieke werking van het algoritme is lastig of niet te achterhalen.
- Werken met grote datasets duurt lang.
- Presteert niet goed met overlappende categorieën.
- Hyperparameter selectie is moeilijk.
- Werkt minder goed met datasets met ruis.

4.6.1.2 DECISION TREES

Decision Trees algoritmen kunnen zowel voor classificatie doeleinden gebruikt worden als voor regressie vraagstukken. De Decision Tree, beslissingsboom, start met een uitgangssituatie (root node) en vertakt in twee mogelijke beslissingen (decision node). Deze beslissingen zijn ook weer knooppunten (nodes) in de beslissingsboom en kunnen weer verder opgesplitst worden. Dit gaat net zolang door totdat er geen data meer aanwezig is om te splitsen, of totdat het vooraf ingestelde maximum aantal splitsingen is bereikt (terminal node of leaf). De node staat voor de beslissing die genomen wordt, de vertakkingen staan voor de combinatie van variabelen die leiden tot deze beslissing en de leaf is de uiteindelijke klasse waarin het te beoordelen element valt. Een voorbeeld van een Decision Tree is te zien in Figuur 15.



FIGUUR 15 VOORBEELD VAN EEN BESLISSINGSBOOM

Decision Trees zijn krachtige algoritmen geschikt voor complexe datasets. Daarnaast zijn ze eenvoudig te begrijpen en te interpreteren en ook eenvoudig in gebruik. Een belangrijk nadeel is dat Decision Trees instabiel kunnen zijn. Instabiliteit betekent hier dat Decision Trees gevoelig zijn voor kleine veranderingen in de dataset met overfitting als gevolg.

Voordelen:

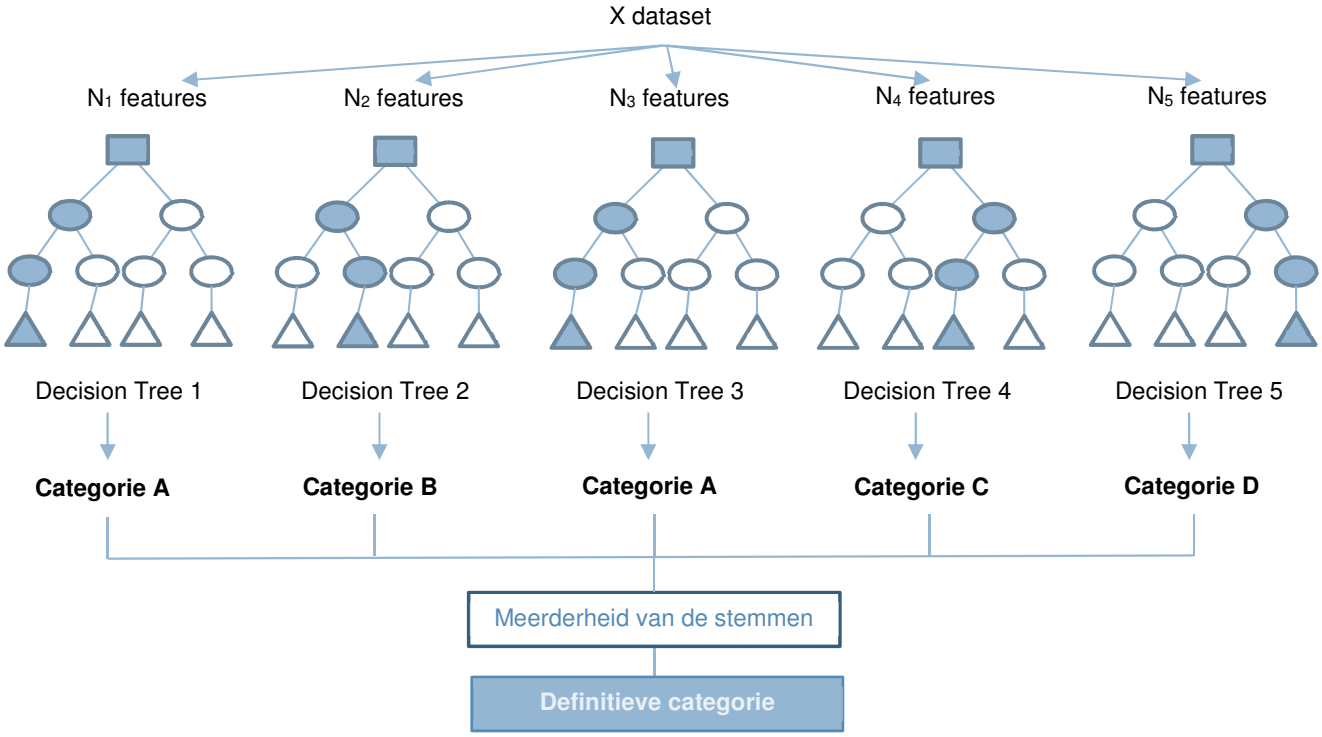
- Eenvoudig te begrijpen en op te zetten.
- Pre-processing van de data is tot een minimum beperkt.
- Snel in runtime: de instructies die worden uitgevoerd terwijl het algoritme wordt uitgevoerd.
- Onbelangrijke features en uitschieters worden automatisch genegeerd.
- Kan overweg met missende data.

Nadelen:

- Leidt snel tot overfitting.
- 'Greedy' algoritme: alleen het huidige beslismoment wordt gewogen, eventuele toekomstige of vorige beslissingen hebben geen invloed meer.
- Voorspellend vermogen is zwak en onstabiel.

4.6.1.3 RANDOM FORESTS

Decision Trees en Random Forests hangen nauw samen: Random Forests is een ensemblemethode in de vorm van meerdere beslisbomen. Deze Decision Trees worden getraind op een willekeurige selectie van variabelen en een even willekeurig deel van de trainingset. Individuele voorspellingen van de Decision Trees worden samen genomen en de categorie met de meeste stemmen bepaalt de definitieve uitkomst, zie Figuur 16.



FIGUUR 16 SCHEMATISCHE WEERGAVE VAN EEN RANDOM FOREST MODEL

Random Forest geeft over het algemeen een betere voorspelling dan een Decision Tree. Ook is bij een Random Forest minder sprake van overfitting.

Voordelen:

- Zeer krachtig algoritme.
- Geeft betere voorspellingen dan Decision Tree.
- Kan overweg met missende data.
- Minder kans op instabiliteit en overfitting.

Nadelen:

- Veel hyperparameters.
- Moeilijk te doorgronden: black box voorspellingsmodel.
- Kost veel geheugen en rekenkracht.

4.6.1.4 NAÏVE BAYES

Het Naïve Bayes (NB) algoritme is een classificatie techniek die gebaseerd is op de Stelling van Bayes. Deze Stelling gaat over de waarschijnlijkheid van een gebeurtenis, gebaseerd op voorkennis van omstandigheden die mogelijk verband houden met de gebeurtenis. Het is een relatief eenvoudig machine learning algoritme dat kansen berekent voor elke mogelijk uitkomst op basis van de bekende waarden van de variabelen. Het model gaat er vanuit dat de gebruikte variabelen, de features, onafhankelijk zijn van elkaar, vandaar het woord "Naïve". In het kort kan Naïve Bayes in de volgende formule worden samengevat:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

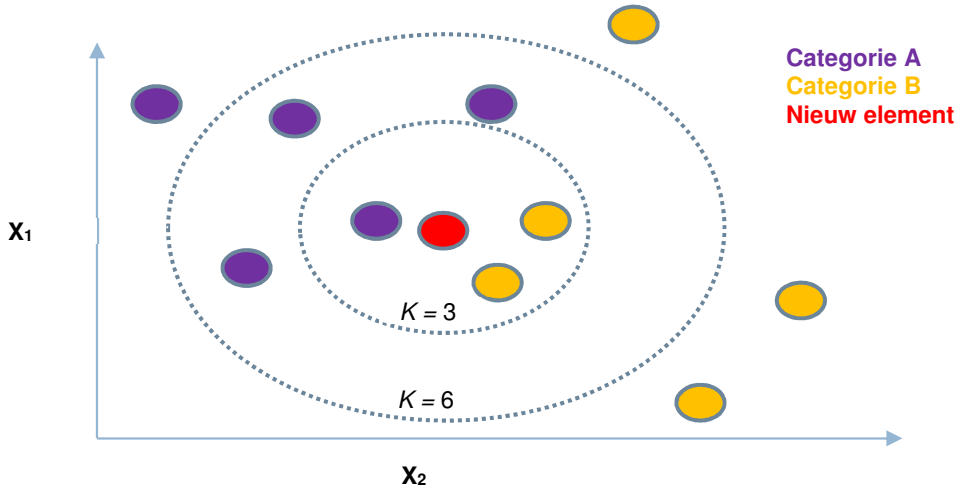
Waar $P(A|B)$ de kans is dat A optreedt gegeven het feit dat B al heeft plaatsgevonden.
Waar $P(B|A)$ de kans is dat B optreedt gegeven het feit dat A al heeft plaatsgevonden.
Waar $P(A)$ de kans is dat A optreedt.
Waar $P(B)$ de kans is dat B optreedt.

Voordelen:
- Robuust algoritme.
- Eenvoudig te implementeren.
- Kan omgaan met missende waarden.
- Het Naïve Bayes algoritme staat bekend als een zeer efficiënt.

Nadelen:
- De sterke aanname dat er geen afhankelijkheid zit tussen de variabelen.
- Kans op verlies van nauwkeurigheid.
- Black box: specifieke werking van het algoritme is lastig of niet te achterhalen.

4.6.1.5 K-NEAREST NEIGHBORS

K-Nearest Neighbors (KNN) is een eenvoudig supervised machine learning algoritmes en wordt zowel voor classificatie als voor regressie gebruikt. Een element wordt geclassificeerd aan de hand van de afstand tot zijn dichtstbijzijnde buren. Waarbij de k verwijst naar het aantal te vergelijken buren van het element. Het kiezen van de juiste waarde hangt onder andere af van de data en hoe deze is verdeeld. Bij classificatie verdient het de voorkeur om een oneven aantal buren te kiezen zodat met meerderheid van stemmen geclassificeerd wordt.
KNN is een 'lazy' algoritme. Dit houdt in dat de parameters van het model niet apart getraind worden. Bij elke voorspelling wordt de gehele trainingset doorlopen op zoek naar de dichtstbijzijnde k-buren. Figuur 17 is een schematische weergave van het KNN algoritme.



FIGUUR 17 SCHEMATISCHE WEERGAVE K-NN

Voordelen:
- Eenvoudig algoritme om te begrijpen en in te zetten.
- Snelle trainingstijd: er hoeft geen model gebouwd te worden. Trainingsfase bestaat alleen uit het opslaan van de features en de klasselabels.
- Maakt geen aannames over onderliggende data.

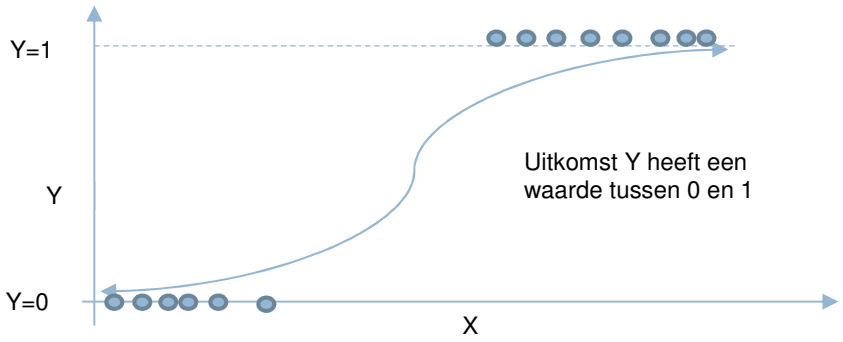
Nadelen:
- Geheugenintensief: alle datapunten moeten worden opgeslagen, dit kost veel rekentijd.
- Gevoelig voor uitschieters.
- Lazy algoritme: de voorspellingsfase kost veel tijd.
- Minder geschikt voor ongebalanceerde datasets.

4.6.1.6 LOGISTIC REGRESSION

De naam doet denken dat het een algoritme is voor regressie vraagstukken, echter dit is niet het geval. Het wordt alleen toegepast bij classificatie vraagstukken en met name bij binaire classificatie problemen. Wanneer het wordt toegepast bij classificatie problemen met meerdere klassen, worden benaderingen gebruikt als één klasse versus de overige klassen.
Bij het gebruik van dit algoritme gaat het erom een verband te vinden tussen kenmerken (features) en de waarschijnlijkheid van een bepaalde uitkomst (in mijn geval: is het domein bonafide of malafide). In de volgende formule wordt dit weergegeven:

$$P(Y = 1|X_1, \ldots . X_p)$$

De kans dat een domein malafide is ($Y = 1$) hangt af van de inputvariabelen X_1 tot en met X_p. De variabelen zijn in dit geval de gekozen features. Het logistic regression algoritme berekent deze kans met behulp van een logistieke functie; de sigmoïd functie (zie Figuur 18). De sigmoïd functie geeft een S-vormige curve die elke input omzet naar een waarde tussen de 0 en 1.



FIGUUR 18 SIGMOÏD FUNCTIE - LOGISTIC REGRESSION

Voordelen:

- Transparant en eenvoudig te interpreteren algoritme.
- Model presteert goed wanneer de dataset lineair te scheiden is.
- Eenvoudig te implementeren en zeer efficiënt te trainen.
- Regularisatie technieken ter voorkoming van overfitting van het model, zijn eenvoudig toepasbaar.

Nadelen:

- Gaat uit van lineaire afhankelijkheid, niet alle features zijn lineair afhankelijk.
- Presteert niet goed als er veel features zijn.
- Kan een groot aantal categorieën per variabele niet goed verwerken.

4.6.2 Performance evaluatie

Het is van belang te weten hoe goed een algoritme presteert. Daarom is het evalueren van machine learning algoritmes een essentieel onderdeel van de modelvorming. Er zijn diverse evaluatie methoden beschikbaar, veelal verdeeld in drie groepen (Hossin & Sulaiman, 2015):

- gebaseerd op drempelwaarden: beoordelen van percentage goed en fout;
- gebaseerd op waarschijnlijkheid: niet alleen beoordelen op percentage goed en fout maar ook berekenen hoe groot de kans is dat een verkeerd antwoord is geselecteerd;
- gebaseerd op rangschikking: beoordelen hoe goed het algoritme de invoer rangschikt.

Elke evaluatie methode benadrukt weer een ander aspect bij het evalueren, waardoor er verschillen in de uitkomsten kunnen ontstaan. Een voorbeeld hiervan is het bepalen van de nauwkeurigheid van een binair classificatie algoritme. Hierbij kan gekeken worden naar vier onderdelen:

- het aantal juist gekwalificeerde voorbeelden die tot de positieve klasse behoren (true positives: tp),
- het aantal juist gekwalificeerde voorbeelden die tot de negatieve klasse behoren (true negatives: tn),
- het aantal onjuist gekwalificeerde voorbeelden die tot de positieve klasse behoren (false positives: fp),
- het aantal onjuist gekwalificeerde voorbeelden die tot de negatieve klasse behoren (false negatives: fn).

Deze vier onderdelen vormen samen de zogenaamde confusion matrix en zijn de basis van de meest gebruikte evaluatie methoden bij binaire classificatie algoritmen (Sokolova & Lapalme, 2009). In Tabel 3 is de confusion matrix weergegeven welke is toegepast op mijn onderzoek.

TABEL 3 CONFUSION MATRIX TOEGEPAST OP MIJN ONDERZOEK

		Voorspelde klasse	
		Negatief	Positief
Daadwerkelijke klasse	Negatief	tn: Domein is bonafide.	fp: Domein is onterecht als malafide gekenmerkt.
	positief	fn: Domein is onterecht als bonafide gekenmerkt.	tp: Domein is malafide.

Elke evaluatie methode heeft een eigen focus. In Tabel 4 is een overzicht gegeven van de meest gebruikte evaluatie methoden en de bijbehorende focus. Tevens is aangegeven wat deze evaluatie methoden betekenen voor mijn onderzoek.

TABEL 4 EVALUATIE METHODEN VOOR BINAIRE CLASSIFICATIE ALGORITMEN GEBASEERD (SOKOLOCA & LAPALME, 2009), (HOSSIN & SULAIMAN, 2005) MET EEN VERWIJZING NAAR MIJN ONDERZOEK

Evaluatie methode	Formule	Evaluatie focus
Accuracy	$\frac{tp + tn}{tp + fn + fp + tn}$	Algehele effectiviteit van een classificatie algoritme: de verhouding tussen het aantal goede voorspellingen en het totaal aantal voorspellingen. Focus binnen mijn onderzoek: hoeveel van de totale voorspellingen van malafide en bonafide domeinnamen zijn juist.
Error rate	$\frac{fp + fn}{tp + fp + tn + fn}$	Misclassificatie: de verhouding tussen het aantal foute voorspellingen en het totaal aantal voorspellingen. Focus binnen mijn onderzoek: hoeveel van de totale voorspellingen van malafide en bonafide domeinnamen zijn onjuist.
Precision	$\frac{tp}{tp + fp}$	Precisie: het juist aantal voorspelde uitkomsten van de positieve klasse door het classificatie algoritme.

		Focus binnen mijn onderzoek: Welke van alle voorspelde malafide domeinnamen zijn daadwerkelijk malafide.
Recall (Sensitivity)	$\frac{tp}{tp+fn}$	Robuustheid: de effectiviteit van een classificatie algoritme om de positieve labels te identificeren. Focus binnen mijn onderzoek: Welk deel van alle malafide domeinnamen is juist voorspeld als malafide.
Fscore	$\frac{(\beta^2+1)tp}{(\beta^2+1)tp+\beta^2 fn+fp}$	De relatie tussen de positieve labels in de dataset en welk deel hiervan door het algoritme als positief wordt voorspeld. Focus binnen mijn onderzoek: welk deel van de malafide domeinnamen worden correct voorspeld en welk deel van alle malafide domeinnamen worden juist voorspeld.
Specificity	$\frac{tn}{fp+tn}$	Hoe effectief voorspelt een classificatie algoritme negatieve labels. Focus binnen mijn onderzoek: welk deel van de bonafide domeinnamen worden correct voorspeld.
AUC (Area under the ROC-curve)	$1/2\,(\frac{tp}{tp+fn}+\frac{tn}{tn+fp})$	Het vermogen van een algoritme om valse classificatie te vermijden. Focus binnen mijn onderzoek: wat is het vermogen van het algoritme om te voorkomen dat ten onrechte wordt voorspeld dat een domeinnaam malafide is of dat ten onrechte wordt voorspeld dat een domeinnaam bonafide is.

Volgens Hossin & Sulaiman en Sokoloca & Lapalme worden in de praktijk vooral evaluatiemethoden gebruikt die gericht zijn op nauwkeurigheid en fout percentages (Hossin & Sulaiman, 2015), (Sokolova & Lapalme, 2009). Ook wel aangeduid met performance en prestatie van het model. Uit mijn literatuuronderzoek komt een ander beeld naar voren, namelijk dat er diverse evaluatiemethoden worden gebruikt. Bovendien worden verschillende namen gebruikt voor dezelfde evaluatiemethoden, of dezelfde naam maar een andere manier van berekenen. Tenslotte worden de gebruikte evaluatiemethoden bij veel onderzoeken niet gespecificeerd. Het vergelijken van onderzoeken op basis van performance indicatoren is dan ook niet goed uit te voeren.
Door zowel Hossin & Sulaiman als Sokolova & Lapalme wordt opgemerkt dat de Accuracy methode niet goed werkt bij ongebalanceerde data. Wanneer een van de beide klassen ernstig ondervertegenwoordigd is kan een vertekend beeld ontstaan. Bijvoorbeeld 97% van de domeinnamen in een dataset van 100 domeinnamen is bonafide en 3% is malafide. Wanneer alle

domeinnamen vervolgens als bonafide worden aangemerkt komt men op een Accuracy percentage van:

$$Accuracy = \frac{true\ positive\ + true\ negative}{true\ positive + false\ negative + false\ positive + true\ negative} = \frac{97 + 0}{97 + 0 + 3 + 0} = \frac{97}{100} = 97\%$$

Hiermee lijkt het alsof het algoritme goed presteert. Echter, juist de afwijkingen, dat wil zeggen de malafide domeinnamen, worden niet herkend. Een oplossing hiervoor is het balanceren van de dataset: zorgen dat elke klasse in gelijke mate is vertegenwoordigd. De F-score is een betere evaluatiemethode voor het berekenen van de nauwkeurigheid van een classificatie algoritme.
In mijn experimenten gebruik ik de evaluatie methoden Precision, Recall, F1 en AUC. De eerste drie geven een beeld hoe goed het model de malafide domeinnamen kan detecteren. De laatste performance indicator, AUC, geeft weer in welke mate het model misclassificatie kan voorkomen. Voor het vergelijken met andere DNS detectiemethoden zal ik de evaluatiemethoden Error rate en Accuracy toevoegen. Met in het achterhoofd de mogelijkheid dat Accuracy een vertekend beeld geeft wanneer de dataset niet gebalanceerd is.

4.7 Conclusie

Machine learning is volop in ontwikkeling. Deze ontwikkeling brengt ook verantwoordelijkheden mee voor ontwikkelaars van machine learning systemen. De gedachte dat ML systemen zelf tot de beste oplossing kunnen komen is mijns ziens onjuist. De ontwikkelaar beïnvloedt (onbewust) het op te zetten machine learning model door de data die hij aanbiedt. Een voorbeeld hiervan is de belastingfraude die het afgelopen half jaar veel in het nieuws is geweest. Een grote groep mensen is onterecht als fraudeur aangemerkt. De opgezette modellen hadden een vooringenomenheid ten opzichte van bepaalde groepen mensen, waardoor deze als fraudeur werden aangemerkt. Deze vooringenomenheid is niet door het model zelf bedacht maar zat al in de aangeboden dataset. Het gevolg was dat het gecreëerde model niet de gewenste patronen kon detecteren. Echter, de uitkomst van dit model is door de belastingdienst, gebruikers ervan, wel als waarheid aangemerkt. Mijns inziens is ML een lerende machine/computersysteem waarbij degenen die een model ontwikkelen een grote invloed hebben op de uitkomsten. Die invloed wordt bepaald door de veronderstellingen die in de data zitten.

Machine learning is één van de meest gebruikte technieken bij botnetdetectie. In mindere mate wordt gebruik gemaakt van deep learning. Botnetdetectie op basis van DNS wordt steeds vaker toegepast omdat tegenwoordig bijna elk botnet gebruik maakt van DNS. Hierbij worden met name clustering en classificatie algoritmen ingezet. Clustering algoritmen worden vooral toegepast wanneer gezocht wordt naar afwijkingen in verkeersstromen en classificatie algoritmen worden vaak ingezet voor het detecteren van malafide domeinnamen.

Voordat ik een definitief machine learning model opstel, worden eerst meerdere algoritmen uitgeprobeerd zonder deze helemaal uit te werken. Op basis van een evaluatie worden de best presterende algoritmen uitgekozen. De uitgekozen algoritmen worden vervolgens tot in detail uitgewerkt: de ML modellen. Om praktische redenen heb ik het aantal uit te proberen algoritmen beperkt tot zes en drie daarvan in detail uitgewerkt.

Een machine learning algoritme presteert het beste wanneer het wordt getraind met een gebalanceerde dataset. Wanneer dit niet het geval is, kunnen evaluatiemethoden een onjuist beeld geven. Dit speelt zeker bij de Accuracy evaluatiemethode. Het lastige is echter, dat juist in de wereld van botnetdetectie in de werkelijkheid data juist vaak niet gebalanceerd is. De verhouding tussen goed verkeer en fout verkeer, of de verhouding tussen bonafide domeinnamen en malafide domeinnamen is niet 50-50. Het overgrote deel is goed verkeer dan wel bonafide domeinnamen en

de afwijking is maar een klein deel. In ons specifieke geval gaat het dan eerder over tiende procenten dan hele procenten.
Het vergelijken van diverse botnetdetectie methoden is ontzettend lastig. Er wordt niet gebruik gemaakt van één universele manier van meten, terwijl de gemeten situatie (omvang van de data, soort data, omgeving waarin data is vastgelegd) ook vrijwel nooit identiek is.

5 DATASET

5.1 Bestaande datasets

Het grootste gedeelte van botnetdetectie technieken is gebaseerd op machine learning, waarbij gebruik gemaakt wordt van datasets. Bij supervised machine learning wordt er gebruik gemaakt van een gelabelde trainingset. Het is dus van belang dat een kwalitatief goede en gelabelde dataset aanwezig is.

Een probleem hierbij is dat er weinig publiekelijk toegankelijke datasets zijn en dat de wel beschikbare datasets al relatief oud zijn (Singh, Singh, & Kaur, 2019), (Ring M. , Wunderlich, Scheuring, Landes, & Hotho, 2019), (Stevanovic & Pedersen, 2013). Door de jaren heen veranderen zowel de karakteristieken van botnet aanvallen als ook het gebruik van internet. Daarom is bij verouderde datasets het gevaar aanwezig dat deze niet meer overeenkomen met de huidige karakteristieken.
Een ander probleem is dat de beschikbare datasets meestal afkomstig zijn van een bedrijfsnetwerk en niet van een Internet Service Provider omgeving (Ring M. , Wunderlich, Scheuring, Landes, & Hotho, 2019), (Zhauniarovich, Khalil, Yu, & Dacier, 2018). Het probleem is dan dat de verkeerskarakteristieken van een ISP daadwerkelijk anders zijn dan die van een bedrijfsnetwerk. Bijvoorbeeld de verhoudingen in soort DNS aanvragen is anders, de diversiteit van opgevraagde domeinnamen zal verschillend zijn maar ook de omvang in aantal actieve hosts zal in een ISP omgeving groter zijn dan in een enterprise omgeving. Omdat mijn bedrijf een lokale ISP is, zal een enterprise gebaseerde dataset voor onze omgeving niet werken.

In Tabel 5 is een overzicht gegeven van op DNS gebaseerde datasets bekend uit de literatuur. Alleen de datasets vanaf 2017 zijn hierin vermeld, dit omdat datasets van vóór 2017 niet meer een reëel beeld geven. De tabel is gebaseerd op onderzoek van onder andere Ring et al., Maciá-Fernández et al. en Damasevicius et al. (Ring M. , Wunderlich, Scheuring, Landes, & Hotho, 2019), (Maciá-Fernández, Camacho, Magán-Carrión, García-Teodoro, & Therón, 2018), (Damasevicius, et al., 2020). Volgens Damasevicius et al. is er de laatste tijd wel een aantal nieuwe netwerk gebaseerde datasets verschenen. Omdat deze nog niet geaccepteerd zijn als benchmark door de wetenschappelijke gemeenschap, zijn ze ook niet in de tabel opgenomen. Singh et al. hebben een overzicht gegeven van gebruikte DNS detectie technieken en de beperkingen van elk van de onderzochte technieken (Singh, Singh, & Kaur, 2019). De onderzochte technieken, of datasets waarop de technieken zijn gebaseerd, zijn samengesteld voor 2017 en zijn daarom ook niet in onderstaande tabel opgenomen.

TABEL 5 OVERZICHT BESTAANDE DATASETS VANAF 2017

Dataset	Datum creatie dataset	Publiekelijk beschikbaar	Legitiem verkeer	Aanval verkeer	Duur	Soort verkeer	Type netwerk	Gelabeld	Toelichting
CIC DoS[1] (Jazi, Gonzalez, Stakhanova, & Ghorbani, 2017)	2017	Ja	Ja	Ja	24 uur	nagebootst	Klein netwerk	Ja	Dataset omvat DoS aanvallen gericht op de applicatie laag gecombineerd met data uit de ISCXIDS2012 dataset. Van de ISCXID2012 dataset is het aanvalsvrije (legitieme) verkeer gebruikt. Dit betekent dat de veranderende netwerk karakteristieken niet opgenomen zijn. Er zijn 4 soorten aanvallen geproduceerd met verschillende tools, wat resulteerde in 8 verschillende DoS-aanvallen vanuit de applicatie laag.
CIC-IDS 2017[2] (Sharafaldin, Lashkari, & Ghorbani, 2018)	2017	Ja	Ja	Ja	5 dagen	Nagebootst	Klein netwerk	Ja	Betreft zowel LAN verkeer als internet communicatie gebaseerd op IDS. Dataset is begin 2020 geupdate. Diverse aanvalssoorten zijn in de dataset verwerkt.
CSE-CIC-IDS2018 on AWS[3] (Sharafaldin, Lashkari, & Ghorbani, 2018)	2018	Ja	Ja	Ja	20 dagen	Nagebootst	Klein netwerk	Ja	Betreft zowel LAN verkeer als internet communicatie gebaseerd op IDS. Diverse aanvalssoorten zijn in de dataset verwerkt. Onder andere ook aanvalsverkeer van binnenuit.
CIDDS-001[4] (Ring M. , Wunderlich,	2017	Ja	Ja	Ja	28 dagen	Nagebootst & productie	Klein netwerk	Ja	Gecombineerde data gebaseerd op IDS: LAN verkeer als ook internet communicatie.

[1] Dataset te vinden op: https://www.unb.ca/cic/datasets/dos-dataset.html
[2] Dataset te vinden op: https://www.kaggle.com/cicdataset/cicids2017
[3] Dataset te vinden op: https://www.unb.ca/cic/datasets/ids-2018.html
[4] Dataset te vinden op: https://www.hs-coburg.de/forschung/forschungsprojekte-oeffentlich/informationstechnologie/cidds-coburg-intrusion-detection-data-sets.html

Gruedl, Landes, & Hotho, 2017)									
CIDDS-002[5] (Ring M. , Wunderlich, Gruedl, Landes, & Hotho, 2017)	2017	Ja	Ja	Ja	14 dagen	Nagebootst	Klein netwerk	Ja	Gecombineerde data op basis van IDS: LAN verkeer als ook internet communicatie.
PUF (Sharma, Singla, & Guleria, 2018)	2018	Onbekend	Ja	Ja	3 dagen	Productie	Universiteit netwerk	ja	Betreft een flow-based DNS dataset. Uit de publicatie is niet te achterhalen of de PUF dataset publiekelijk toegankelijk is.
TRAbID[6] (Viegas, Santin, & Oliveira, 2017)	2017	Ja	Ja	Ja	8 uur	Nagebootst	Klein netwerk	Ja	Gecombineerde data op basis van IDS: LAN verkeer als ook internet communicatie.
Unified Host and Network[7]	2017	Ja	Ja	Ja	90 dagen	Productie	Enterprise netwerk	nee	Betreft een subset van het netwerkverkeer van het Los Alamos National Laboratory. Het betreft zowel LAN verkeer als internet communicatie.
CIRA-CIC-DoHBrw-2020[8]	2020	Ja	Ja	Ja	onbekend	Nagebootst	Klein netwerk	ja	Dataset richt zich specifiek op getunneld kwaadaardig DNS verkeer (DoH). Dataset bevat zowel non-DoH HTTPs verkeer als normaal DoH en kwaadaardig DoH verkeer. Kwaadaardig non-DoH is niet aanwezig.

[5] Dataset te vinden op: https://www.hs-coburg.de/forschung/forschungsprojekte-oeffentlich/informationstechnologie/cidds-coburg-intrusion-detection-data-sets.html
[6] Dataset te vinden op: https://secplab.ppgia.pucpr.br/?q=trabid
[7] Dataset te vinden op: https://csr.lanl.gov/data/2017/
[8] Dataset te vinden op: https://www.unb.ca/cic/datasets/dohbrw-2020.html

5.2 Opzet eigen dataset

Voor mijn onderzoek heb ik DNS data nodig die ik kan verrijken met GEO informatie en met informatie over de ouderdom van opgevraagde domeinnamen. Kenmerk van een malafide domeinnaam is dat deze maar een beperkte levensduur heeft. Dit betekent dat bij datasets die meer dan een jaar oud zijn, de levensduur van een malafide domeinnaam niet meer achterhaald kan worden. Ook worden bij lang niet alle datasets de opgevraagde domeinen vastgelegd. De GEO informatie van IP-adressen kan eveneens in de loop van de tijd gewijzigd zijn omdat IPv4 adressen van eigenaar kunnen veranderen.
Ook heb ik voor mijn onderzoek een dataset nodig die gebaseerd is op een ISP omgeving, deze is lastig te verkrijgen. Daarom heb ik gekozen een eigen dataset te creëren op basis van data afkomstig uit ons eigen ISP netwerk.

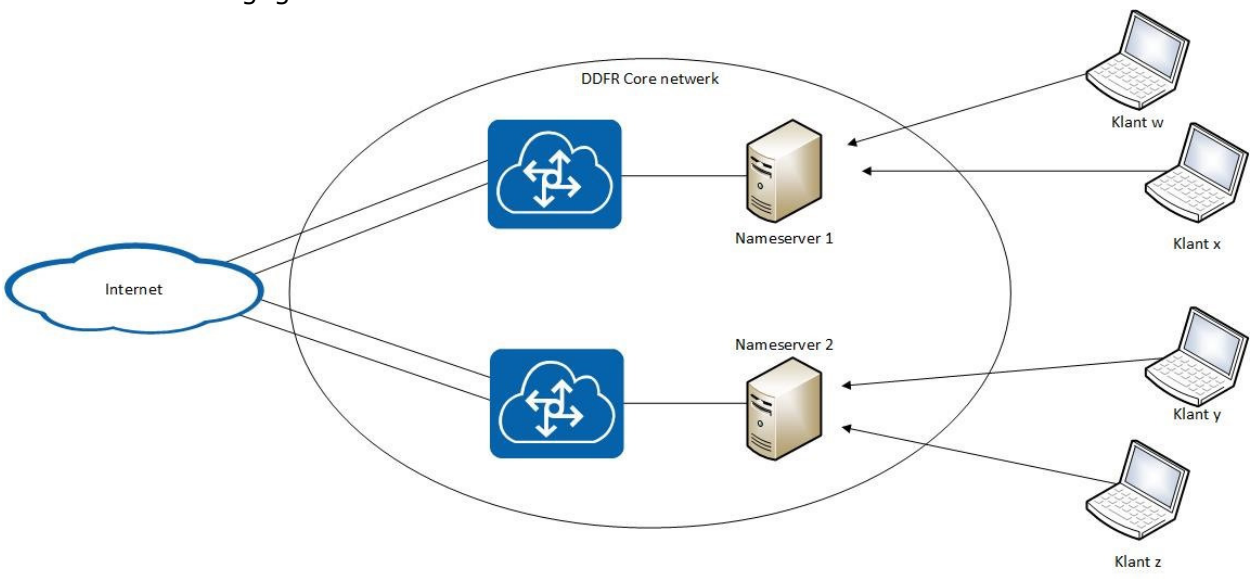
5.2.1 Datacollectie

Voor het onderzoek is een offline opstelling gemaakt: data wordt eerst opgeslagen en daarna ingelezen in de testopstelling. Deze opstelling voor het detecteren van botnets heeft geen directe invloed op de productieomgeving. Een online opstelling geeft daarentegen mogelijk risico's op verstoring in de productie omgeving van mijn bedrijf, wat vanuit bedrijfsoogpunt niet acceptabel is.

In deze paragraaf zal worden beschreven in welke omgeving de data zijn verzameld, gevolgd door een beschrijving van de dataset zelf. Ik maak hierbij gebruik van de dataset beschrijving zoals opgesteld door Ring et al. (Ring M. , Wunderlich, Scheuring, Landes, & Hotho, 2019). Ring et al. heeft een framework opgesteld dat diverse dataset eigenschappen beschrijft. Doel hiervan is om datasets met elkaar te kunnen vergelijken. Het framework is door Ring et al. gebaseerd op de 4 principes van Wilkinson waar wetenschappelijke data aan moet voldoen: vindbaarheid, toegankelijkheid, herbruikbaarheid en interoperabiliteit (voor meerdere doeleinden geschikt) (Wilkinson, 2016). Hoewel mijn dataset niet publiekelijk beschikbaar is, kan met deze beschrijving door andere onderzoekers wel een dergelijke dataset worden gerealiseerd en gebruikt worden voor nieuw of aanvullend onderzoek.

Het netwerk

DDFR is een lokale Internet Service Provider (ISP) in Friesland met alleen bedrijfsmatige klanten. De internetaansluiting is 30GB met een gemiddelde load van 20GB. In Figuur 19 is dit schematisch weergegeven.



FIGUUR 19 NETWERKTEKENING DDFR

Data eigenschappen
De data, afkomstig van onze DNS servers, uit de periode van 01-12-2020 tot en met 09-12-2020 is vastgelegd en heeft een omvang van 700GB. Het betreft ruwe data in de vorm van PCAP files, welke zowel bestaat uit DNS query's als DNS answers. Voor mijn onderzoek zijn alleen DNS answers nodig. Deze answers bevatten de benodigde informatie voor het opstellen van mijn features. Uiteindelijk is uit praktische overwegingen (doorlooptijd, rekenkracht) er voor gekozen om van één dag data de malafide domeinnamen te selecteren en een subset van bonafide domeinnamen te nemen om zo te komen tot een dataset.

5.2.2 Features

Botnets maken steeds vaker gebruik van zowel domain-flux technieken als fast-flux technieken om ontdekking en ontmanteling tegen te gaan. Zoals eerder aangegeven in hoofdstuk 3 zijn domain-flux en fast-flux elkaar tegenovergestelde. Bij domain-flux is juist het domein wat snel wisselt en niet de IP-adressen en bij fast-flux juist de IP-adressen en niet het domein. Wanneer beide technieken gebruikt worden, kunnen kenmerken van beide technieken worden gebruikt als feature die voor elk afzonderlijk kenmerkend zijn.

Op basis van literatuuronderzoek naar het DNS protocol (paragraaf 3.4), fast-flux en domain-flux (paragraaf 3.5) en DNS gebaseerde botnetdetectie methoden (paragraaf 4.5.2), ben ik tot de volgende selectie van tien features gekomen:

- **Aantal IP-adressen:** het aantal IP-adressen zal meer dan gemiddeld zijn bij flux botnets. Grote cloud providers gebruiken ook meerdere IP-adressen maar domeinnamen met weinig IP-adressen kunnen worden uitgesloten.
- **Msglen**: lengte van het DNS anwser bericht. Mogelijk dat malafide DNS answers een afwijkende lengte hebben dan reguliere DNS answers.
- **ASN:** mogelijk dat binnen bepaalde AS nummers verhoogde botnet activiteit is waar te nemen.
- **Aantal AS-nummers:** IP-adressen van domain-flux en fast-flux botnets zullen verdeeld zijn over meerdere AS-nummers. Dit zal bij legitiem gebruik beperkter zijn.
- **ASN registrar:** mogelijk dat bij bepaalde AS registrars verhoogde botnet activiteit waar te nemen is.
- **Aantal ASN Registrars:** over het algemeen zijn legitieme IP-adressen behorend bij een domeinnaam terug te brengen naar één ASN registrar. Als meerdere ASN registrars betrokken zijn, kan dit een indicatie zijn voor malafide gebruik.
- **Landen:** botnetinfecties zijn in bepaalde landen meer aanwezig dan in andere landen. Het land van herkomst kan een indicatie zijn voor een grotere kans dat een domeinnaam malafide is.
- **Aantal landen:** IP-adressen van domain-flux en fast-flux botnets zullen meer verspreid zijn over diverse landen. Bij grote cloud-providers die wereldwijd hun diensten aanbieden, ziet men ook wel spreiding van IP-adressen maar deze zal meer gegroepeerd zijn per land/regio.
- **Ouderdom domeinnaam:** domeinnamen die gebruikt worden bij domain-flux zijn maar kort actief. Deze zullen dus aanzienlijk jonger zijn dan bonafide domeinnamen. Dit kenmerk zal een sterk aanwijzend karakter hebben. De andere features hebben vooral een aanvullend bewijzend karakter.
- **TTL:** een korte TTL wordt gebruikt bij fast-flux. Dit wordt zowel door botnets gebruikt als op legitieme wijze. Deze feature is vooral bedoeld als scheiding: domeinnamen met een hoge TTL zullen niet malafide zijn.

In Tabel 6 zijn een aantal voorbeelden te zien van waarden van de gekozen features.

TABEL 6 GEKOZEN FEATURES MET AANTAL VOORBEELDEN VAN BIJBEHORENDE WAARDEN

#IP adressen	Msglen	ASN	#ASN	Registrar	#Registrars	Landen	#Landen	Ouderdom domeinnaam	TTL	Malafide
1	190	16625	1	arin	1	Netherlands	1	6998	2	0
3	86	13335	1	arin	1	United_States	1	326	202	1
2	79	16276	1	ripencc	1	Poland, France	2	1462	3600	1
1	148	60341	1	ripencc	1	Netherlands	1	8875	120	0
1	82	4766	1	apnic	1	South_Korea	1	8527	1800	1
1	49	35467	1	ripencc	1	Netherlands	1	5467	8104	0
1	673	48635	1	ripencc	1	Netherlands	1	6204	172800	0
1	1033	63949	1	arin	1	United_States	1	NULL	60	1
1	60	32651 396549 396557 396566 396574 397197 397203	7	arin	1	United_States	1	9282	3519732	0
4	102	12876, 5577, 40824, 200350	4	ripencc, arin	2	United_States, Luxembourg, France, Russia	4	9282	3519732	0

Door Singh et al. en Zhauniarovic et al. is onderzocht welke features zijn toegepast in de diverse onderzoeken. Hieruit blijkt dat het gebruiken van features zowel op basis van DNS answers als op basis van dataverrijking met Geo informatie ook in andere onderzoeken wordt toegepast. De feature ASN registrar en Country worden niet specifiek door hen besproken en ook in de onderzochte methoden ben ik deze twee niet tegen gekomen (Singh, Singh, & Kaur, 2019), (Zhauniarovich, Khalil, Yu, & Dacier, 2018).
De door mij gekozen combinatie van features ben ik niet tegengekomen in de onderzochte DNS detectiemethoden. Mijn gekozen aantal features is relatief klein te noemen in verhouding tot de onderzochte methoden (paragraaf 4.5.1, Tabel 1).

5.2.3 Verwerking data & balanceren dataset

Om de PCAP files te kunnen verwerken is DNSanon gebruikt (stap A in Figuur 20). DNSanon is een tool om DNS verkeer van PCAP files om te zetten naar tekst (The Ant Lab, 2019). Optioneel kan gekozen worden voor anonimiseren. Er is voor gekozen om niet te anonimiseren omdat in de uiteindelijke dataset geen privacy gevoelige informatie aanwezig is. Na omzetting met DNSanon is de privacy gevoelige informatie verwijderd.
DNSanon creëert drie files van elke PCAP: request-, message- en rr-file. Request files bevatten domein opvragen en worden in mijn onderzoek niet gebruikt. Message file is het antwoord op een domein opvraag, inclusief de timestamp van de opvraag en de message lengte. Alleen de messagelengte wordt in mijn onderzoek gebruikt. In de rr-file zitten naast de opgevraagde domeinnaam aanvullende gegevens zoals ANcount, NScount, ARcount en TTL. Deze informatie is nodig voor de verdere verwerking. De omvang van de benodigde files bedraagt ongeveer 450GB. Eén dag data levert ongeveer 350 miljoen records op in de database. Dit zijn geen unieke domein requests want diverse domeinen worden meerdere malen opgevraagd. Het aantal unieke domein opvragen ligt rond de 450 duizend. De benodigde files worden ingelezen in een SQLDB (stap B in Figuur 20) .

Om de verkregen data te labelen worden vervolgens twee stappen parallel uitgevoerd. Alle domeinnamen van 01-12-2020 worden naar Tesorion gestuurd waar de blacklisting plaats vindt. Voor whitelisting wordt de Tranco-lijst gebruikt (stap C in Figuur 20). In paragraaf 5.2.4 wordt het labeling proces nader toegelicht.

Van de 450 duizend unieke domeinnamen waren er uiteindelijk 233 malafide volgens de analyse van Tesorion. Dit is een kleine 0,07% van het geheel. Het deel malafide domeinnamen is dus

maar een heel klein deel van de totale data voor deze ene dag. Deze dag is echter wel representatief qua verkeershoeveelheid. Het is dan ook aannemelijk dat andere dagen een soortgelijk resultaat opleveren.
Voor machine learning algoritmen is het gewenst dat beide categorieën evenredig voorkomen in de dataset (Krawczyk, 2016), (Viloria, Lezama, & Mercado-Caruzo, 2020). Zoals aangegeven in paragraaf 4.6.2 met betrekking tot de performance evaluatie, kan een algoritme een succespercentage van 99,93% behalen door het negeren van de minderheidsklasse (de malafide domeinen) en alle domeinen als bonafide te classificeren. Echter, hiermee wordt juist de afwijking, de malafide domeinnamen, onjuist geclassificeerd. Het blijkt dat juist in veel praktijksituaties de verhouding tussen de verschillende klassen, of de distributie van de voorbeelden, niet gebalanceerd is (Krawczyk, 2016). Onze praktijksituatie is hierop geen uitzondering. Om goede resultaten met machine learning te kunnen halen, zal de data dus uiteindelijk moeten worden gebalanceerd.
Er zijn drie manieren om datasets te balanceren: dataset aanpassen, algoritme aanpassen of beide mogelijkheden gecombineerd (Krawczyk, 2016). Wel geeft Krawczyk aan dat deze methoden uitgaan van ongebalanceerde datasets met een verhouding van 1:4 tot 1:100. In de praktijk worden verhoudingen gezien van 1:1000 tot 1:5000. Het is onduidelijk hoe goed deze methoden werken bij zulke extreem ongebalanceerde datasets. Daar is aanvullend onderzoek voor nodig (Krawczyk, 2016).

Ik heb in mijn onderzoek gekozen voor het aanpassen van de dataset in plaats van het algoritme. Dit betekent concreet dat van de minderheidsklasse voorbeelden gedubbeld moesten worden (oversampling). En voor de meerderheidsklasse zullen voorbeelden verminderd moeten worden (undersampling). Waarbij opgemerkt moet worden dat de nu bekende methoden voor oversampling en undersampling niet geschikt zijn voor deze extreem ongebalanceerde data. Omdat de minderheidsklasse zo ondervertegenwoordigd is, is de kans groot dat de data in deze klasse niet uniform verdeeld is. De karakteristieken en kenmerken van deze klasse moeten wel behouden blijven (Krawczyk, 2016). Helaas viel het onderzoeken hoe under- en oversampling technieken kunnen worden aangepast, buiten mijn onderzoek.

Idealiter zou ik de tien dagen aan data willen analyseren en verrijken en vervolgens de stappen nemen van undersampling en oversampling. De doorlooptijd van het verwerken van deze data inclusief de verrijking, is dermate lang dat dit niet haalbaar is.
De lange doorlooptijd wordt enerzijds veroorzaakt door de grote hoeveelheid data. Dergelijke grote hoeveelheden data kosten veel rekenkracht en wanneer die niet in voldoende mate aanwezig is, is de benodigde rekentijd lang. Anderzijds nemen de bewerkingen om data te verrijken ook veel tijd. Gratis mogelijkheden hiervoor zijn vaak gelimiteerd. Wanneer alle data op deze wijze verrijkt zou worden, zou dit maanden duren. Bovendien is er een reële kans dat de data niet meer aanwezig is omdat een domeinnaam inmiddels niet meer bestaat. In de literatuur zien we dit probleem ook terug komen. Zo constateren Zhauniarovic et al. dat het opzetten en onderhouden van een dataset waarbij verrijking nodig is, veel middelen vraagt en dat deze lang niet altijd beschikbaar zijn voor onderzoekers (Zhauniarovich, Khalil, Yu, & Dacier, 2018).

Alle 350 miljoen opgevraagde domeinnamen (het veld Name van sectie Answer, zie Figuur 8 DNS MESSAGE ONDERVERDEELD IN SEGMENTEN) van 01-12-2020 zijn opgestuurd naar Tesorion en door hen geanalyseerd. Op basis van deze analyse konden 2867 unieke DNS answer messages, dus met een uniek ID in de Header sectie, als malafide worden aangemerkt. Uit deze messages kon maar een zeer klein aantal unieke malafide domeinnamen worden gedestilleerd: 233. Hierop heb ik besloten om op twee manieren naar de data te kijken, wat geleid heeft tot twee verschillende datasets:

- **DatasetI - DDFR DNS:** Deze dataset bevat alle unieke malafide DNS answers messages. De malafide domeinnamen zijn niet uniek. Dit betekent dat een malafide domeinnaam

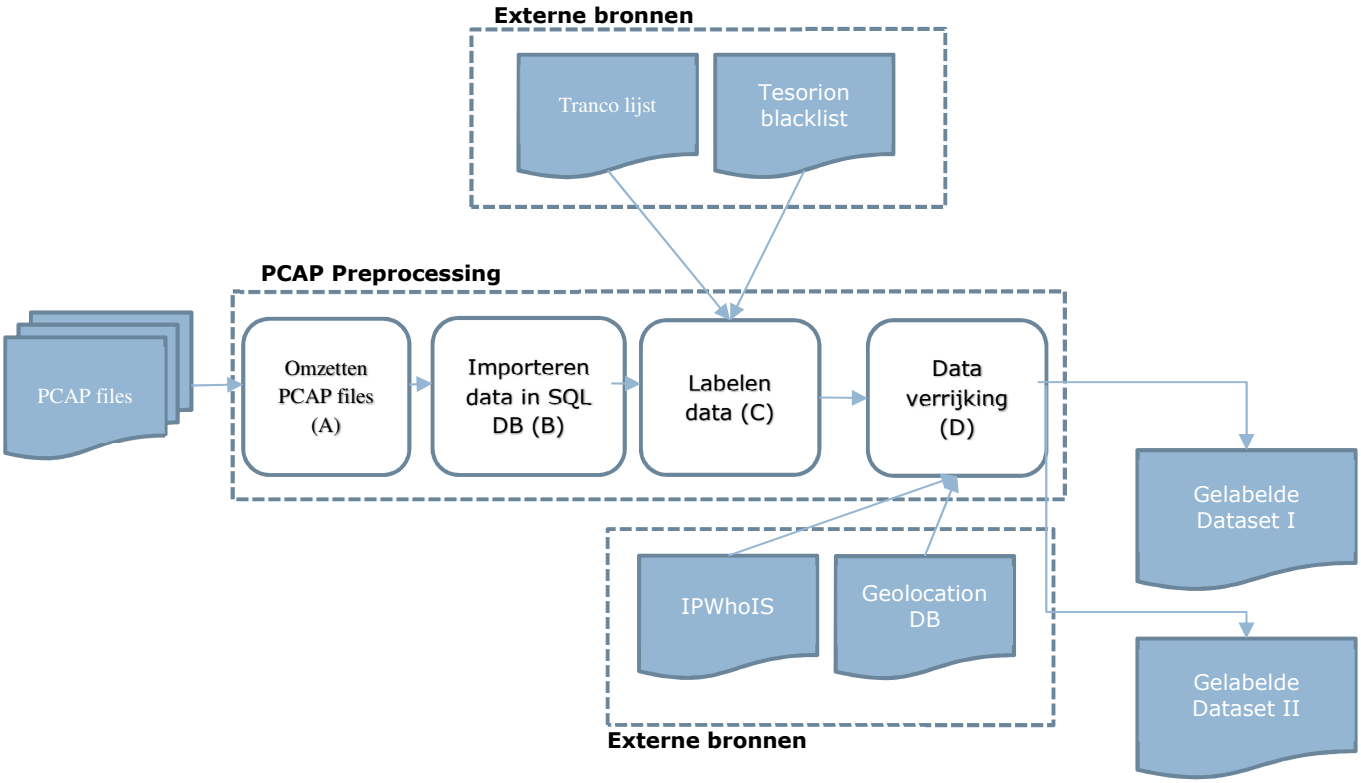
meerdere keren op 01-12-2020 is opgevraagd bij onze DNS server. Bij een deel van de DNS answer messages behorend bij één malafide domeinnaam is het Rdata veld verschillend: verschillende IP-adressen zijn als antwoord terug gegeven bij eenzelfde malafide domeinnaam. DatasetI bevat 2867 malafide DNS answers messages waarvan 233 unieke domeinnamen.
De bonafide domeinnamen zijn allemaal wel uniek: elke bonafide domeinnaam betreft een unieke bonafide DNS answer message. Van vijf minuten data zijn domeinen, welke niet als malafide gelabeld zijn, gevalideerd met de Tranco-lijst. De Tranco-lijst bevat alleen hoofddomein namen. Subdomeinen van deze bonafide domeinnamen zijn eveneens als bonafide gelabeld. De verhouding in DatasetI is nu als volgt: 2511 bonafide en 2867 malafide domeinnamen.

- **DatasetII – DDFR Domains:** Deze dataset bevat zowel unieke malafide domeinnamen als unieke bonafide domeinnamen. Alle malafide DNS answer messages van 01-12-2020 zijn per domein samengevoegd. Per feature is dat als volgt uitgevoerd:
    - Aantal IP-adressen: alle verschillende IP-adressen in de DNS answers zijn bij elkaar opgeteld per malafide domeinnaam. Op gelijke wijze is dit uitgevoerd voor Aantal AS nummers, Aantal Registrars en Aantal landen.
    - Msglen: alle waarden van Msglen van de DNS answers per malafide domeinnaam zijn bij elkaar opgeteld om vervolgens het gemiddelde hiervan te berekenen. Op gelijke wijze is dit uitgevoerd voor TTL.
    - ASN: alle verschillende voorkomende ASN's in de DNS answers per domeinnaam zijn vastgelegd. Op gelijke wijze is dit gedaan voor de features ASN Registrar en Landen.
    - Ouderom domeinnaam: de waarde is voor elk DNS answer gelijk omdat het allemaal domeinanswers betreffen van dezelfde dag. De waarde wordt vastgelegd op basis van de eerst tegengekomen DNS answer van een malafide domeinnaam.

  DatasetII bevat nu 233 malafide domeinnamen. Dezelfde bonafide domeinnamen worden gebruikt als in DatasetI: 2511 bonafide domeinnamen.

Naast het labelen moeten beide datasets verrijkt worden met GEO informatie, ASN en met de ouderdom van de domeinnaam (stap D in Figuur 20). Hiervoor is gebruik gemaakt van de bronnen IPWhois en Geolocation DB. IPWhois is een package beschikbaar via de python community en via Github. In de Github community is het de bekendste IP Whois packages (op basis van rating: aantal stars = 442). De package bestaat al meerdere jaren, wordt onderhouden en de laatste versie is van september 2020. Met deze package zijn bijbehorend ASN en ASN registrar van een IP-adres achterhaald en de ouderdom van een domeinnaam. Via de api Geolocation DB, actief sinds 2018, is vervolgens het land achterhaald waar een IP-adres vandaan komt. Uit statistieken van deze api is alleen het gebruik van de huidige maand te zien, het gebruik varieert tussen de 50.000 en 150.000 aanvragen per dag (Hane, 2020), (Geolocation DB, 2020).
Er heeft enige tijd gezeten tussen het vastleggen van de data (01-12-2020) en het verrijken van de data (januari). Mogelijk dat hierdoor niet alle verrijkingsslagen meer konden worden uitgevoerd of andere informatie geeft dan wanneer deze in december zou zijn uitgevoerd. Bijvoorbeeld omdat een domeinnaam niet meer bestond of dat een IP-adres van eigenaar is gewisseld. De verwachting is dat de impact hiervan nihil is, omdat het een korte periode betreft.

In Figuur 20 is het opstellen van beide datasets schematisch weergegeven. Het betreft hier de ruwe dataset: verzamelde data welke gebruikt wordt als input voor het uitvoeren van de experimenten.



FIGUUR 20: SCHEMATISCHE WEERGAVE OPBOUW DATASETS

5.2.4 Bepalen ground truth

Een van de vele uitdagingen bij het opzetten van een dataset is het bepalen van de 'ground truth'. Wat betekent dit? Ground truth wil zeggen fundamentele waarheid, de fundamentele waarde van het te detecteren element. In mijn onderzoek gaat het om het bepalen of een domeinnaam bonafide of malafide is. Eigenlijk is de naam ground truth misleidend. Het geeft het idee dat van een dataset voor elk element onomstotelijk vastgesteld kan worden wat de juiste uitkomst is. Dat is niet het geval. Immers de samensteller van de dataset bepaalt wat de juiste uitkomst moet zijn en bepaalt hiermee het ideale systeemgedrag. Ground truth is hierbij niet een onomstotelijk vastgesteld gegeven. Het is goed te beseffen dat een machine learning model op basis van een ground truth getraind en/of geëvalueerd wordt. Eventuele voorkeuren in een dataset worden hiermee overgenomen door een machine learning model.

Dit betekent dat de uitkomst van het ML model sterk afhankelijk is van de gronden waarop besloten wordt of een domeinnaam malafide of bonafide is. Het is mogelijk om op basis van de beschikbare informatie een beslissing te nemen over de vraag of een domeinnaam bonafide of malafide is. Echter, de informatie waarop deze beslissing genomen wordt, wordt ook gebruikt als feature in het model. Bijvoorbeeld, de ouderdom van een domeinnaam: botnets gebruiken maar zeer kort domeinnamen en de registratiedatum zal dan ook zeer recent zijn. Als op basis hiervan gelabeld wordt, is het de vraag of het model nog wel objectief met deze feature kan omgaan. Er is een waarde toegekend aan deze informatie en het model 'erft' deze waarde en zal deze verder versterken. Het is dus nodig de domeinnamen te labelen met informatie die niet als feature in het model gebruikt wordt.

In de literatuur is terug te vinden dat whitelists en blacklist veel worden toegepast bij het vaststellen van de ground truth (Singh, Singh, & Kaur, 2019), (Bilge, Sen, Balzarotti, Kirda, & Kruegel, 2014), (Silva, Silva, Pinto, & Salles, 2012), (Stevanovic M. , Pedersen, D'Alconzo, Ruehrup, & Berger, 2015). Punt van aandacht is dat wanneer er verschillende blacklists met elkaar vergeleken worden, er weinig overlap is. Dit blijkt uit onderzoek van Jhaveri et al. (Jhaveri,

Cetin, Gañán, & Van Eeten, 2017). Een van de oorzaken hiervan kan zijn dat blacklists verschillende criteria hebben om te bepalen of een domeinnaam op de blacklist komt (Stevanovic M. , Pedersen, D'Alconzo, Ruehrup, & Berger, 2015), (Singh, Singh, & Kaur, 2019). Daarnaast kan het ook zo zijn dat een blacklist gewoonweg niet alles heeft kunnen waarnemen wat kwaadaardig is.

Voor het bepalen van de ground truth van een dataset, waarbij gebruik gemaakt wordt van een blacklist, heeft de voorkeur om verschillende blacklists te gebruiken. En te beoordelen, indien mogelijk, welke criteria worden gebruikt of wat voor soort botnets / malware, aanwezig zijn in de blacklist. In de praktijk zal dit lastig zijn en worden de domeinen op de blacklist als een gegeven beschouwd.

Naast de blacklists worden vaak de whitelists gebruikt. Bekende varianten hiervan zijn Alexa en Tranco (Alexa, 2020), (Tranco, 2021). Whitelists zijn lijsten met de meest populaire domeinen waarbij vanuit gegaan wordt dat domeinen die populair zijn ook goedaardig zijn. Whitelists kunnen worden gebruikt om de hoeveelheid data die beoordeeld moet worden te verkleinen. Domeinen die aanwezig zijn op deze lijst, worden gelabeld als bonafide en hoeven niet nader onderzocht te worden.

Voor mijn onderzoek heb ik gekozen om de domeinnamen te laten beoordelen door Tesorion, een Nederlands cybersecurity bedrijf. Tesorion biedt een dienst aan om DNS verkeer te beoordelen waar ik gebruik van heb mogen maken. Zij hebben alle domeinnamen van 01-12-2020 ontvangen en beoordeeld welke hiervan malafide zijn. Tesorion kan alleen aangeven of een domein kwaadaardig is wanneer het betreffende botnet reeds bij hen bekend is. Tesorion maakte bij het labelen gebruik van meerdere bronnen. Welke bronnen dit zijn, mocht niet worden prijsgegeven. Ook is onbekend welke algoritmen door de bronnen zijn gebruikt. Tesorion staat als security bedrijf goed aanschreven, is onder meer associate partner van het 'NoMoreRansom' project en heeft ook binnen Microsoft de status van 'Thread Indicator Top Contributor'. Toch kunnen er 'false negatives' in de lijst aanwezig zijn. Een extra check bij andere partijen, zoals bijvoorbeeld Virustotal, zou zeker de voorkeur hebben. Echter, gezien de omvang van de brondata en de doorlooptijd die een dergelijke check kost, is dat voor dit onderzoek buiten beschouwing gelaten. Daarnaast werden de domeinnamen gevalideerd met de Tranco-lijst. De Tranco-lijst is ontwikkeld door Le Pochat et al. om de onderzoeksgemeenschap te kunnen laten werken met betrouwbare en reproduceerbare rankingslijsten. Tranco is gebaseerd op de 4 meest bekende rankinglijsten waaronder Alexa, maar is minder eenvoudig te manipuleren. Ook wisselen de domeinnamen in de lijst minder snel en waardoor reproduceerbaarheid van een onderzoek met deze lijst eenvoudiger is (Le Pochat, Van Goethem, Tajalizadehkoob, Korczynski, & Joosen, 2019).
Aanvullend is een lijst lokale domeinnamen handmatig gecontroleerd en aangemerkt als bonafide domeinnamen. Aldus is de ground truth bepaald. Wanneer domeinnamen op Tranco-lijst voorkwamen én door Tesorion gelabeld waren als malafide, zijn ze als malafide opgenomen in de dataset.

In Tabel 7 zijn de eigenschappen van deze twee datasets samengevat volgens het framework van Ring et al. (Ring M. , Wunderlich, Scheuring, Landes, & Hotho, 2019).

TABEL 7 BESCHRIJVING DATASET VOOR MIJN ONDERZOEK VOLGENS FRAMEWORK RING ET AL.

Dataset eigenschappen DatasetI: DDFR_DNSREQUESTS & DatasetII: DDFR_DOMAINS		
Algemene informatie	Jaar van samenstellen	2020
	Publiekelijk beschikbaar	Nee
	Normaal verkeer	Ja: dataset bevat domeinnamen die bonafide zijn.
	Kwaadaardig verkeer	Ja: dataset bevat domeinnamen die kwaadaardig zijn. Onder andere van de volgende botnets zijn domeinnamen opgenomen: QSnatch, Pitou , Rovnix, Pushdo.B en diverse malware varianten zoals Trojan.Crypt.
Soort data	Metadata	De dataset bestaat uit de volgende kolommen: - Domeinnaam: domeinnaam die opgevraagd is. Wanneer de dataset geprepareerd wordt voor het model, wordt deze kolom verwijderd. - TTL: time to live, tijd in seconden waarbinnen een opgevraagde domeinnaam uit de cache van een DNS server gehaald kan worden. Datatype is een integer en kan bij beide datasets variëren tussen 0 en 3519732 seconden. - Msglen: lengte van het DNS bericht. Datatype is een integer en kan bij DatasetI variëren tussen 44 en 1668 bytes en bij DatasetII tussen 44 en 8214 bytes. Bij DatasetII is het gemiddelde genomen van de waarden van de verschillende answers behorend bij één malafide domeinnaan. - ASN: AS nummer(s) behorend bij (het) IP-adres(sen) die terug gegeven is/zijn bij opvraag domeinnaam. Datatype is een object. - ASN Registrar: de na(a)m(en) van de registrar(s) behorend bij (een) AS nummer(s). Datatype is een object. - Countries: (het) land(en) waar (het) IP-adres(sen) vandaan kom(t)(en). Datatype is een object. - Ip_count: het aantal IP-adressen dat terug gegeven is bij opvraag domeinnaam. Datatype is een float64 en kan bij beide datasets variëren tussen 1 en 17. - ASN_count: het aantal verschillende AS nummers behorend bij de IP-adressen die terug zijn gegeven bij opvraag domeinnaam. Datatype is een float64 en kan variëren tussen 1 en 8 voor beide datasets. - ASN_reg_count: het aantal verschillende ASN registrars behorend bij de AS nummers behorend bij de opvraagde domeinnaam. Datatype is een float64 en kan variëren tussen 1 en 2 voor beide datasets. - Country_count: het aantal verschillende landen waar de IP-adressen vandaan komen, de IP-adressen zijn de IP-

		adressen die terug zijn gegeven bij opvraag domeinnaam. Datatype is een float64 en kan variëren tussen 1 en 8 voor beide datasets. - Domain_days_old: hoe lang bestaat het domein. Datatype is een float64 en kan variëren tussen 9 dagen en 12692 dagen voor beide datasets. - Malicious: geeft aan of de domeinnaam bonafide is of malafide. Deze kolom is het label waarbij de waarde nul staat voor bonafide en één voor malafide. Datatype is een integer. De dataset is gevalideerd met behulp van Tesorion voor het bepalen van de kwaadaardige domeinnamen en met behulp van de Tranco-lijst, gedownload op 04 januari 2021, voor de bonafide domeinnamen. Tot slot zijn honderdvijfendertig bij ons veel voorkomende, bonafide domeinnamen toegevoegd en handmatig geverifieerd omdat ze niet in de Tranco-lijst voorkomen. GEO informatie is gewonnen met behulp van de volgende bron: https://geolocation-db.com ASN informatie en creatie datum van een domeinnaam (om ouderdom mee te berekenen) is gewonnen met behulp van de volgende bronnen: https://pypi.org/project/ipwhois https://www.sidn.nl/whois
	Format	Uiteindelijke dataset is een csv file. Bron zijn PCAP files.
	Anonimiteit	Dataset is anoniem. Bevat geen IP-adressen van de bron. Alleen publiekelijk bereikbare domeinnamen en bijbehorende publiekelijk beschikbare informatie.
Data Volume	Hoeveelheid	DatasetI: 2511 unieke bonafide domein answers, elke request is tevens een uniek domeinnaam. 2867 unieke malafide domein answers, van de 2867 unieke malafide domein answers zijn er 233 unieke malafide domeinnamen. Alle malafide DNS answers zijn in de dataset opgenomen. DatasetII: 2511 unieke bonafide domein answers, elke request is tevens een uniek domeinnaam. 233 malafide domeinnamen waarbij de informatie van de verschillende answers is geaggregeerd.
	Duur	Uiteindelijk zijn alle malafide domeinnamen tijdens de duur van één dag vastgelegd: 01-12-2020 en van de bonafide domeinnamen is een subset genomen van vijf minuten van 01-12-2020.
Recording Environment	Soort verkeer	Passive DNS data: alleen de answers. Data is verrijkt met GEO informatie, AS nummer, ASN registrar en ouderdomsgegevens van het domeinnaam.
	Type netwerk	Kleine lokale ISP met ongeveer 25000 gebruikers.
	Compleet netwerk	Ja, data is afkomstig van een productie DNS server.
Evaluatie	Verhouding training / test	Beide datasets zijn opgesplitst in de volgende verhouding: 80% voor training en 20% voor testdoeleinden.

	Balanced	DatasetI: dataset is in verhouding en hoeft niet te worden gebalanceerd. DatasetII: dataset is niet in verhouding en zal moeten worden gebalanceerd.
	Labeled	Ja, labeling is gedaan op grond van blacklisting en whitelisting (Le Pochat, Van Goethem, Tajalizadehkoob, Korczynski, & Joosen, 2019). Voor whitelisting is gebruik gemaakt van de Tranco-lijst (Tranco, 2021) gedownload op 04 januari 2021. Bij blacklisting is gebruik gemaakt van de diensten van het bedrijf Tesorion en voor de 135 handmatig toegevoegde domeinnamen is ook nog gebruik gemaakt van Virustotal (www.virustotal.com).

5.3 Conclusie

De basis voor botnetdetectie is een betrouwbare dataset die gelabeld is en gebruikt kan worden voor training en validatie. Er zijn echter maar relatief weinig datasets publiekelijk beschikbaar en nog minder gelabelde datasets. Elke dataset heeft zijn eigen insteek waardoor niet elke dataset voor elk onderzoek geschikt is. Vooraf is ook niet altijd duidelijk hoe een dataset is opgebouwd. Het framework van Ring et al. helpt bij het beoordelen van een dataset en ook bij het beschrijven van een zelf samen te stellen dataset (Ring M. , Wunderlich, Scheuring, Landes, & Hotho, 2019). De meeste datasets zijn minimaal meerdere jaren oud. Voor mijn onderzoek was het nodig om data te verrijken met GEO informatie en ouderdom informatie. Deze aanvullende data is lastig terug te vinden wanneer een dataset wat ouder is. Dit alles heeft geleid tot mijn besluit een eigen dataset samen te stellen.

Uit mijn literatuuronderzoek is gebleken dat de features Landen, Aantal landen, ASN Registrar en Aantal ASN Registrars niet eerder zijn gebruikt bij DNS gebaseerde botnetdetectie methoden. De overige features zijn al wel eerder gebruikt. Mijn combinatie van gebruikte feature is nog niet eerder toegepast.

Het bepalen van de ground truth zorgt voor een belangrijk deel hoe een machine learning model presteert. Het model kan bij een testset wel een juiste score hebben, maar in de praktijk toch niet goed presteren. Bijvoorbeeld omdat een domeinnaam niet op de juiste manier gelabeld is. Voor het bepalen van de ground truth van mijn model was het niet mogelijk meerdere blacklisten te gebruiken.

In mijn dataset was de verhouding tussen malafide en bonafide domeinnamen ongebalanceerd: 1:1400. Machine learning algoritmen kunnen hier niet goed mee omgaan. Het balanceren van de dataset is hierbij een vereiste. De huidige methoden gaan bij een ongebalanceerde dataset uit van een verhouding tussen de 1:4 tot 1:100 en niet van extremen zoals 1:1400. Er is nog veel onderzoek nodig hoe hier mee om te gaan. Het lastige is dat juist in de praktijk deze extremen vaak voorkomen. In reeds bekende datasets ziet men dit nauwelijks terug. Hooguit een opmerking dat de dataset is gebalanceerd.

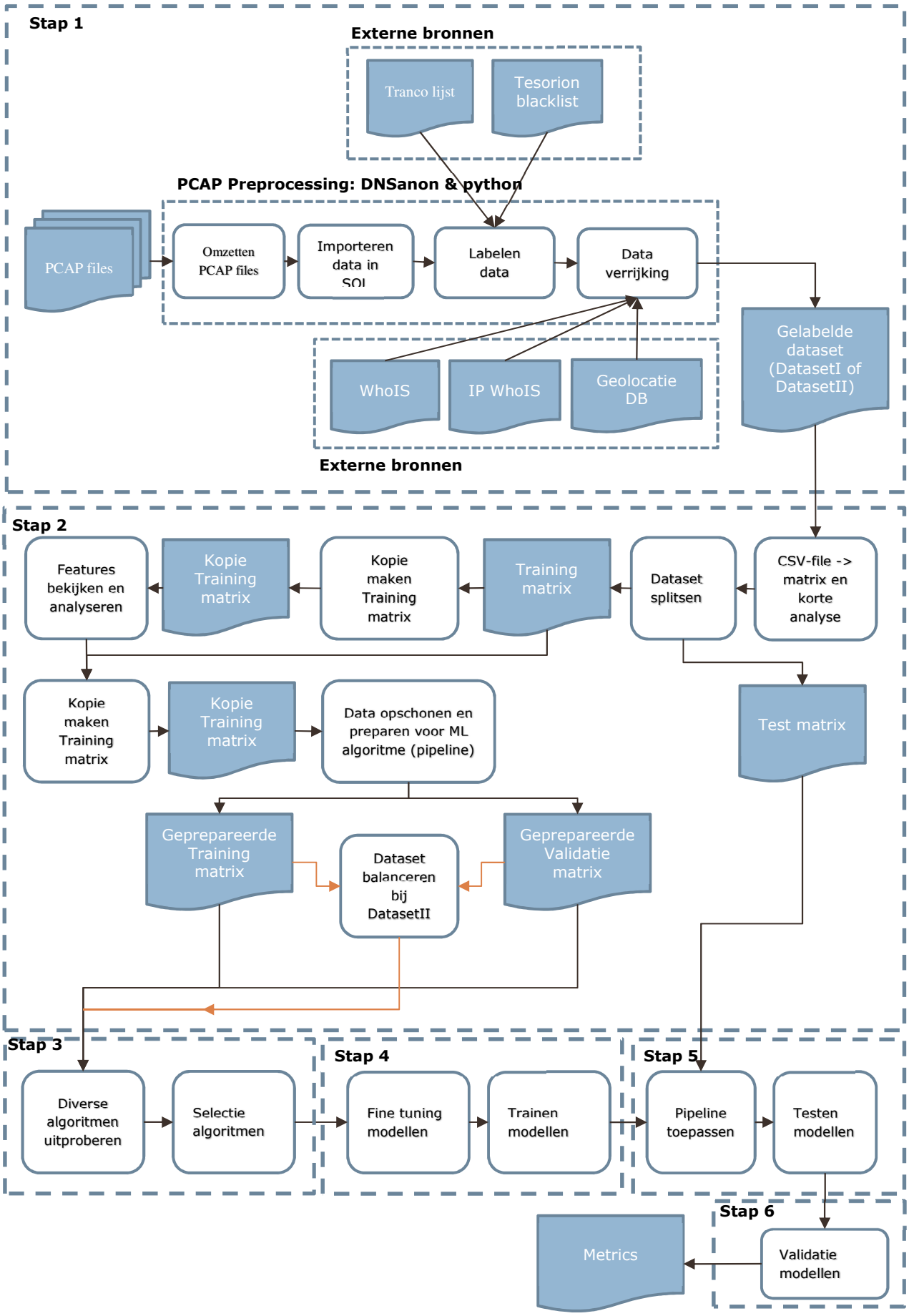
Voor het toepassen van undersampling en oversampling van de dataset is het nodig dat eerst de verrijking van de data wordt gedaan. Deze verrijkingsslagen kosten naast veel rekenkracht ook zeer veel doorlooptijd omdat gratis bronnen die hiervoor gebruikt werden maar beperkt beschikbaar zijn. Gevolg zou zijn dat domeinen dan niet meer bestaan of IP-adressen van eigenaar gewisseld zijn en daarmee de informatie onjuist of onvolledig. Als oplossing heb ik er voor gekozen om niet alle data van 01-12-2020 te nemen maar slechts vijf minuten voor de bonafide domeinnamen en alle DNS requests van de malafide domeinnamen van de hele dag.

6 OPZET EN RESULTATEN VAN DE EXPERIMENTEN

In hoofdlijnen bestaat een machine learning proces uit de volgende 7 stappen:

1. Het verzamelen van data.
2. Data voorbereiden.
3. Opzetten van een concept model.
4. Een model trainen.
5. Een model testen.
6. Een model valideren.
7. Een model in productie nemen.

In stap één, toegelicht in hoofdstuk 5, wordt de dataset samengesteld. Deze wordt in stap twee gesplitst in een training- en testset (paragraaf 6.16.2). Daarna wordt de trainingsdata geprepareerd. Dit houdt onder andere in dat wordt beslist hoe met ontbrekende waarden om te gaan, of een dataset gebalanceerd moet worden en hoe de verhoudingen tussen de features zijn. Indien de trainingset scheef gebalanceerd is, moet dit eerst worden hersteld.
In stap drie (paragraaf 6.2) worden met geprepareerde trainingsets diverse algoritmes getraind zonder te kijken naar de specifieke instellingen (hyperparameters) per algoritme. Op basis van de uitkomsten heb ik een aantal algoritmes gekozen, namelijk de drie best presterende algoritmen.
In stap vier (paragraaf 6.3) wordt voor elk van deze algoritmes een model opgezet waarbij de juiste waarden worden gezocht voor de bijbehorende hyperparameters. De modellen worden getraind om ze vervolgens in stap vijf (paragraaf 6.4) te testen met data die de modellen nog niet eerder hebben gezien. Met de testresultaten kunnen de modellen worden gevalideerd: stap zes (paragraaf 6.5).
Op basis van de validatie wordt het best presterende model gekozen om in productie te nemen. Deze laatste stap, stap zeven, valt buiten het kader van mijn afstudeeropdracht. In Figuur 21 heb ik de stappen één tot en met zes van het machine learning proces schematisch weergegeven.



FIGUUR 21 SCHEMATISCHE WEERGAVE MACHINE LEARNING PROCES

6.1 Data voorbereiden

In paragraaf 5.2.3 is beschreven dat en waarom er twee datasets zijn gemaakt. De verschillen tussen de twee datasets zijn:

- In DatasetI DDFR-DNS (in de rest van dit hoofstuk aangegeven met DatasetI) is elke DNS answer op een malafide domeinvraag (request) vastgelegd in de dataset. Malafide domeinnamen zijn niet uniek. De DNS antwoorden (anwsers) zijn wel uniek.
- In DatasetII DDFR-Domains (in de rest van dit hoofstuk aangegeven met DatasetII) zijn alle DNS answers op een malafide domein request samengevoegd. Malafide domeinnamen zijn uniek. Er is een gemiddelde waarde genomen van de TTL en de msglen. De overige waarden zijn bij elkaar opgeteld in het geval van numerieke waarden dan wel samengevoegd in het geval van tekstuele waarden (objecten).
- De klasse malafide domeinen is in DatasetII ondervertegenwoordigd. Met behulp van SMOTE NC, in paragraaf 6.1.2 nader toegelicht, wordt is de dataset gebalanceerd.

6.1.1 Machine Learning Framework

Voor het uitvoeren van de experimenten is een machine learning tool nodig. Er zijn vele verschillende varianten aanwezig. Om praktische redenen heb ik drie tools met elkaar vergeleken: Weka, RapidMinder en Scikit-Learn.

Weka
Weka is ontwikkeld vanuit de Universiteit van Waikator, Nieuw-Zeeland, en is een open source framework. Sterke punten van Weka zijn dat het framework breed inzetbaar is, veel algoritmen ondersteunt, evenals de standaard taken zoals feature selectie, clustering, classificatie, regressie en visualisatie (Nguyen, et al., 2019).

RapidMiner
RapidMiner, voorheen YALE: Yet Antother Learning Environment, is ontwikkeld in 2001 door Mierswa et al. aan de Technische Universiteit van Dortmund en is een platform onafhankelijk framework (Mierswa, Klinkenberg, Fischer, & Ritthof). Dit framework ondersteunt veel algoritmen, is breed inzetbaar en heeft een grote community. Tegenwoordig is het een commercieel machine learning framework. Voor academische toepassingen kan men een educatieve licentie aanvragen.

Scikit-Learn
Scikit-Learn is ontwikkeld door David Cournapeau in 2007 en is ook een open source framework dat specifiek is ontwikkeld voor machine learning applicaties in de programmeertaal Python (Wang, Liu, Li, Zhu, & Zhang, 2019). Scikit-learn wordt vooral onderhouden en doorontwikkeld door leden van een grote en actieve community. Op basis van de toegankelijkheid en de vele voorbeelden op internet heb ik gekozen voor Scikit-Learn. Wanneer de data van alle 10 dagen zou worden gebruikt, zou Scikit-Learn minder snel zijn vergeleken met andere frameworks, dit is in mijn onderzoek niet van toepassing.

Naast het framework gebruik ik Jupyter notebooks als programmeer omgeving en de server waar de experimenten op uitgevoerd worden, heeft de volgende specificaties:12 vCPUs, 64GB memory en opslag van 4,5TB.

6.1.2 Matrix, korte analyse en dataset splitsen

Een algoritme presteert het meest optimaal wanneer de te gebruiken dataset aan bepaalde eigenschappen voldoet: klassen die evenredig verdeeld zijn (gebalanceerd), een dataset die geen null-waarden (lege velden) bevat, waarden van features die bij voorkeur tussen de 0 en 1 liggen en/of een vergelijkbare schaal hebben en features die een normaal verdeling hebben. Een

normaal verdeling, ook wel gauss-verdeling genoemd, is symmetrisch geconcentreerd rond een centrale waarde. Afwijkingen van deze centrale waarde komen steeds minder voor naar mate de afwijking groter is. Een dataset zal in de werkelijkheid niet aan (al) deze voorwaarden voldoen. Vandaar dat na omvormen van de data naar matrix format, een eerste korte analyse is gedaan om te bepalen welke stappen uiteindelijk moeten worden genomen om de data klaar te maken voor het model (zie paragraaf 646.1.4).

De beide datasets zijn vastgelegd in een csv-file en ingelezen in Jupyter, elk in een eigen omgeving (notebook). De csv-files zijn omgezet naar matrixformaat. Figuur 22 is een weergave van de eerste 5 rijen van de matrix van DatasetI. Een soortgelijke weergave kan voor DatasetII gemaakt worden. Elke rij representeert een domeinnaam en de bijbehorende gegevens. Elke kolom van de matrix is een feature met uitzondering van de kolom 'malicious', deze kolom bevat de labels en geeft bij '1' aan dat het domein malicious is en bij '0' dat het domein bonafide is.

	ttl	msglen	asn	asn_reg	countries	ip_count	asn_count	asn_reg_count	country_count	domain_days_old	malicious
0	1685	80	49	arin	United_States	1.0	1.0	1.0	1.0	NaN	0
1	1707	58	3598	arin	United_States	1.0	1.0	1.0	1.0	NaN	0
2	6780	59	35467	ripencc	Netherlands	1.0	1.0	1.0	1.0	NaN	0
3	11	141	15703	ripencc	Netherlands	6.0	1.0	1.0	1.0	NaN	0
4	1680	80	49	arin	United_States	1.0	1.0	1.0	1.0	NaN	0

FIGUUR 22 EERSTE VIJF RIJEN VAN DATASETI: DDFR DNS

In Tabel 8 is een beschrijving per dataset gegeven: welke velden (kolommen in de dataset) zijn aanwezig, wat is het type en hoe vaak komt het veld voor. In deze tabel zijn de velden die null-waarden bevatten blauw gekleurd.

TABEL 8 BESCHRIJVING VELDEN IN DE BEIDE DATASETS

	Verdeling Dataset		**DatasetI**	**DatasetII**
	Totaal aantal rijen in dataset		5378	2744
	Aantal bonafide domeinnamen		2511	2511
	Aantal malafide domeinnamen		2867	233
Kolomnaam in matrix	**Toelichting**	**DataType**	**Aantal niet-lege velden**	**Aantal niet-lege velden**
Ttl	Time to live	int64	5378	2744
Msglen	Lengte bericht	int64	5378	2744
Asn	ASN nummer	object	*5356*	*2724*
asn_reg	ASN registrar	object	*5358*	*2726*
countries	Land waar IP adres vandaan komt	object	*5204*	*2574*
ip_count	Aantal IP-adressen behorend bij domeinnaam	float64	*5377*	*2743*
asn_count	Aantal ASN nummers behorend bij domeinnaam	float64	*5358*	*2726*
asn_reg_count	Aantal registrars behorend bij de IP-adressen behorend bij een domeinnaam	float64	*5358*	*2726*
country_count	Aantal landen	float64	*5204*	*2574*
domain_days_old	Ouderdom van een domeinnaam	float64	*4822*	*2368*
malicious	Label of domeinnaam malafide is of niet.	int64	5378	2744

Voordat de datasets opgedeeld worden in een trainingset en een testset, wordt een samenvatting gemaakt van de numerieke features, ook wel de basis statistische metrieken

genoemd. In Figuur 23 is dit voor DatasetI weergegeven en in Figuur 24 voor DatasetII. De rijen in het overzicht betekenen het volgende:

- Count: totaal aantal waarden dat aanwezig is per feature. Dit kan verschillen wanneer sommige features lege velden bevatten. Bijvoorbeeld bij domain_days_old zijn er maar 4822 waarden gevonden, terwijl in totaal 5378 rijen in de dataset aanwezig zijn.
- Mean: de gemiddelde waarde van al deze rijen voor de betreffende feature. Zo heeft in DatasetI een domein gemiddeld 2,19 IP-adressen per domein en is een domein gemiddeld 4020 dagen oud.
- Std: de standaarddeviatie, welke de spreiding van de waarden aangeeft.
- Min en max: de minimum waarde en de maximum waarde.
- 25%, 50% en 75%: deze rijen tonen een waarde dat een bepaald percentage van de groep heeft. Bijvoorbeeld de kolom msglen: 25% van alle waarden is gelijk of kleiner dan 74.

	ttl	msglen	ip_count	asn_count	asn_reg_count	country_count	domain_days_old	malicious
count	5.378000e+03	5378.000000	5377.000000	5358.000000	5358.000000	5204.000000	4822.000000	5378.000000
mean	6.684058e+03	144.877464	2.193602	1.058231	1.021650	1.045542	4020.099129	0.533098
std	5.522813e+04	148.614064	1.731377	0.371123	0.145551	0.286209	3155.099783	0.498950
min	0.000000e+00	44.000000	1.000000	1.000000	1.000000	1.000000	9.000000	0.000000
25%	6.000000e+01	74.000000	1.000000	1.000000	1.000000	1.000000	516.000000	0.000000
50%	3.000000e+02	92.000000	2.000000	1.000000	1.000000	1.000000	4032.000000	1.000000
75%	6.000000e+02	148.000000	3.000000	1.000000	1.000000	1.000000	6998.000000	1.000000
max	3.519732e+06	1668.000000	17.000000	8.000000	2.000000	8.000000	12692.000000	1.000000

FIGUUR 23 SAMENVATTING VAN DE NUMERIEKE FEATURES VAN DATASETI: DDFR DNS

	ttl	msglen	ip_count	asn_count	asn_reg_count	country_count	domain_days_old	malicious
count	2.744000e+03	2744.000000	2743.000000	2726.000000	2726.000000	2574.000000	2368.000000	2744.000000
mean	7.567345e+03	189.629738	2.297849	1.060528	1.023478	1.069542	5470.256757	0.084913
std	7.609090e+04	314.647720	2.270996	0.421884	0.151443	0.368056	2525.919656	0.278802
min	0.000000e+00	44.000000	1.000000	1.000000	1.000000	1.000000	9.000000	0.000000
25%	3.000000e+01	79.000000	1.000000	1.000000	1.000000	1.000000	3518.000000	0.000000
50%	6.000000e+01	124.000000	1.000000	1.000000	1.000000	1.000000	5584.000000	0.000000
75%	3.600000e+02	223.000000	3.000000	1.000000	1.000000	1.000000	7874.000000	0.000000
max	3.519732e+06	8214.000000	17.000000	8.000000	2.000000	8.000000	12692.000000	1.000000

FIGUUR 24 SAMENVATTING VAN DE NUMERIEKE FEATURES VAN DATASETII: DDFR DOMAINS

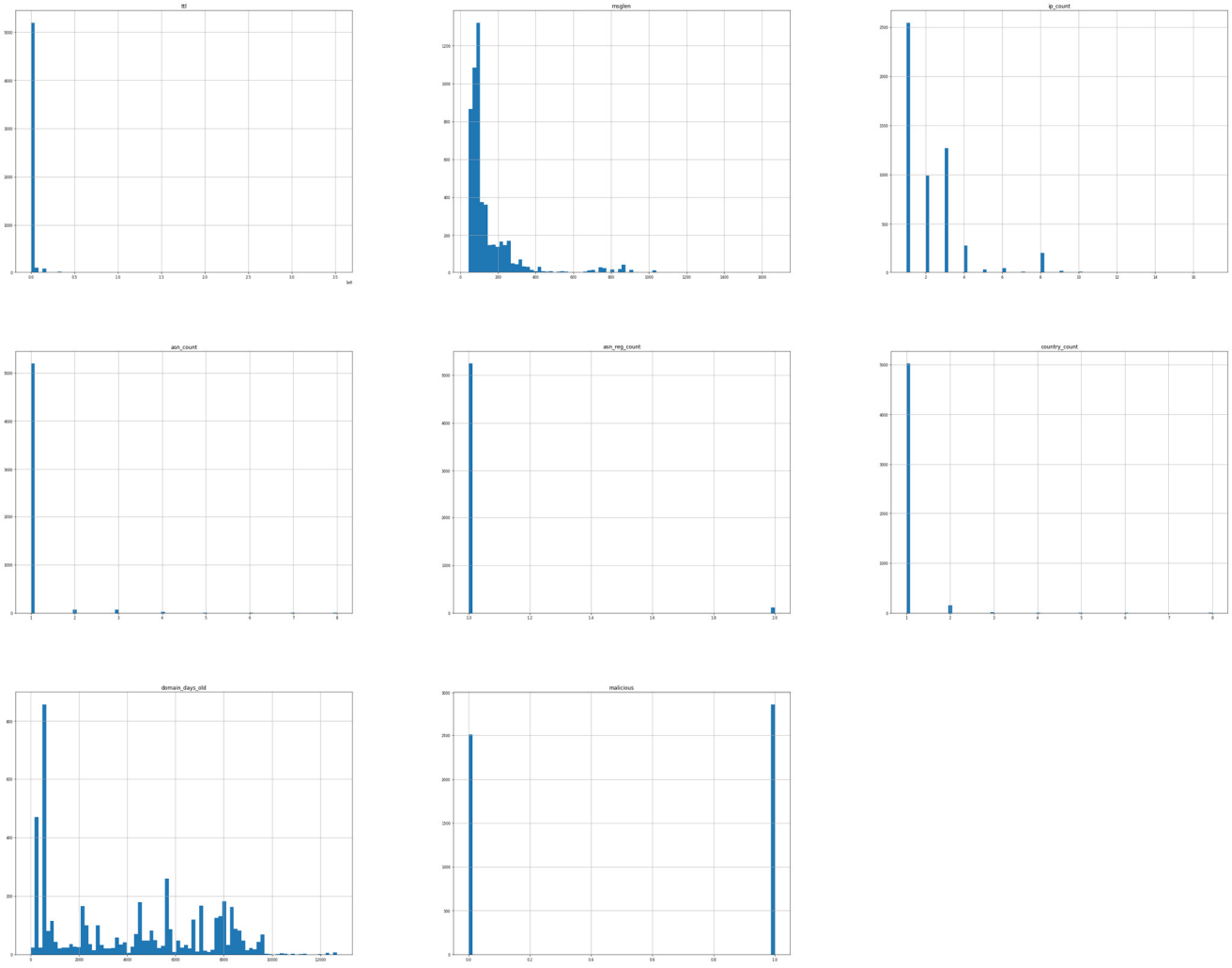
Uit Figuur 23 en Figuur 24 zijn onder andere de volgende verschillen in de datasets te destilleren:

- DatasetI is ongeveer twee maal zo groot in omvang als DatasetII.
- Bij DatasetI ligt het gemiddelde bij malicious op 0.53, dit houdt in dat er iets meer malicious domeinnamen aanwezig zijn dan bonafide. Bij DatasetII is het gemiddelde 0.08, dit betekent dat deze dataset ongebalanceerd is en malicious domeinnamen weinig voorkomen.
- De invloed van het minder aanwezig zijn van malicious domeinnamen zien we ook terug bij de feature domain_days_old: de gemiddelde ouderdom ligt hoger.

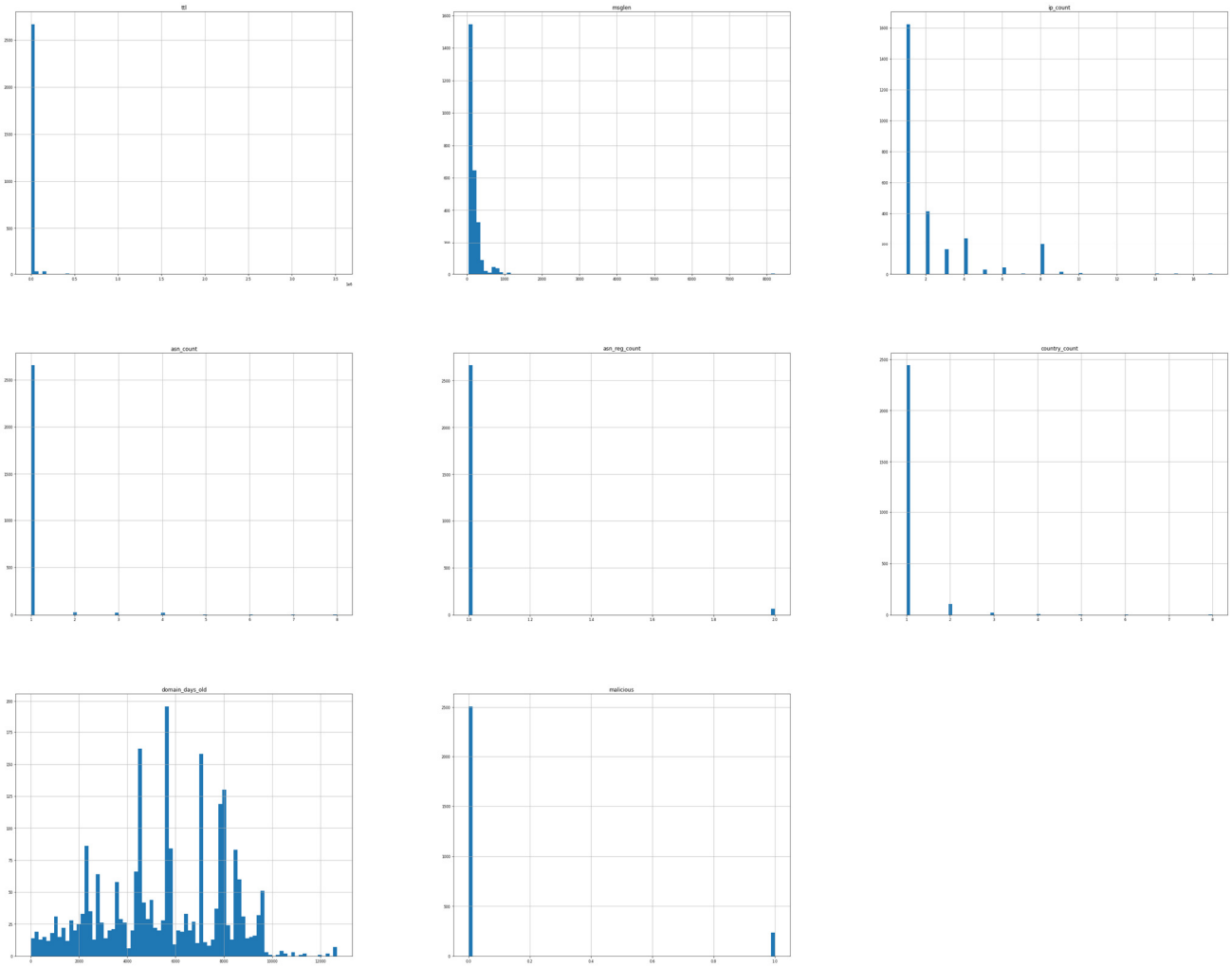
Met histogrammen kan ook inzicht worden verkregen hoe de numerieke features zijn verdeeld. Tevens geeft dit inzicht in de vraag of de dataset gebalanceerd is of niet. In een histogram worden per numerieke feature de waarden gegroepeerd en wordt aangegeven hoeveel van elke groep aanwezig is. Wanneer een feature een normaal verdeling heeft, komt dat tot uiting in een belvormige verdeling, zoals bijvoorbeeld bij de feature 'domain_days_old', zie Figuur 25 en Figuur 26 . Features kunnen ook exponentieel verdeeld zijn, zoals de features 'ttl', 'msglen', 'ip_count', 'asn_count' en 'country_count', zie Figuur 25 en Figuur 26. De feature

'asn_reg_count' heeft net als het label 'malicious' twee waarden. Bij DatasetI is goed te zien dat zowel de bonafide als de malafide domeinen bijna gelijk verdeeld zijn. Bij DatasetII is de malafide klasse ernstig ondervertegenwoordigd.
Beide figuren analyserend, komen de volgende interessante punten naar voren:

- Op basis van de theorie is de aanname gedaan dat de ttl bij malafide domeinen aanzienlijk korter is dan bij bonafide domeinen. Echter, uit histogram 'ttl' kan worden opgemaakt dat de ttl in bijna alle voorkomende gevallen kort is. De feature 'ttl' werkt eerder als een whitelist argument: wanneer de ttl lang is, is het aannemelijk dat het domein bonafide is.
- Histogram 'domain_days_old' laat bij DatasetI aanzienlijk meer 'jonge' domeinen zien dan bij DatasetII. DatasetI bevat ook aanzienlijk meer malafide domeinen dan DatasetII. Aanname is dat jonge domeinen vooral malafide domeinen zijn. De feature 'Domain_days_old' is in dit geval een goede indicator voor malafide domeinen.
- De spreiding bij 'asn_count', 'asn_reg_count' en 'country count' is zeer beperkt. Bij de meerderheid van de domeinen behorende de bijbehorende IP-adressen tot één ASN nummer, één ASN registrar en komen de IP-adressen uit één land. Deze lijken op basis van onderstaande histogrammen, minder relevant.
- De spreiding in de waarden van de feature 'msglen' is bij DatasetI groter dan bij DatasetII. Aanname is dat de spreiding in message lengte groter is bij malafide DNS answers messages.
- Histogram 'IP_count' van beide datasets vergelijkend, zien we bij beide een zelfde soort spreiding terwijl bij DatasetII de malafide domeinen ernstig ondervertegenwoordigd zijn. Mogelijk dat feature 'IP_count' minder goed voorspellend is dan vooraf aangenomen.



FIGUUR 25: HISTOGRAMMEN VAN SPREIDING VAN DE FEATURES DATASETI (V.L.N.R.): TTL, MSGLEN, IP_COUNT, ASN_COUNT, ASN_REG_COUNT, COUNTRY_COUNT, DOMAIN_DAYS_OLD, MALICIOUS.



FIGUUR 26 HISTOGRAMMEN VAN SPREIDING VAN DE FEATURES DATASETII (V.L.N.R.): TTL, MSGLEN, IP_COUNT, ASN_COUNT, ASN_REG_COUNT, COUNTRY_COUNT, DOMAIN_DAYS_OLD, MALICIOUS

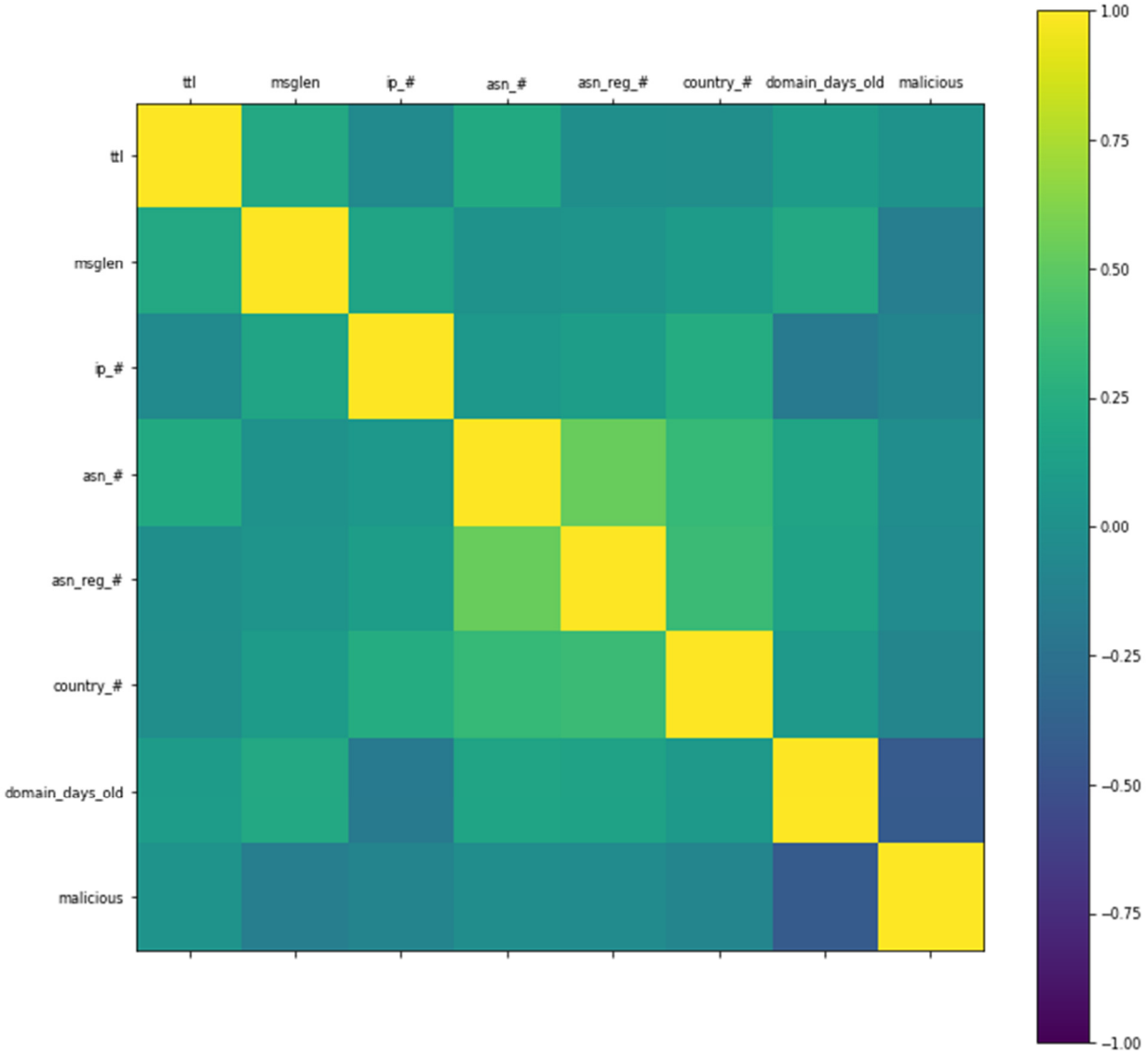
Voordat ik verder ben gegaan met de analyse, heb ik eerst de dataset gesplitst in een trainingset en een testset en een kopie gemaakt van de trainingset. Indien nodig, kan altijd nog terug gevallen worden op de oorspronkelijke trainingset.

6.1.3 Analyse

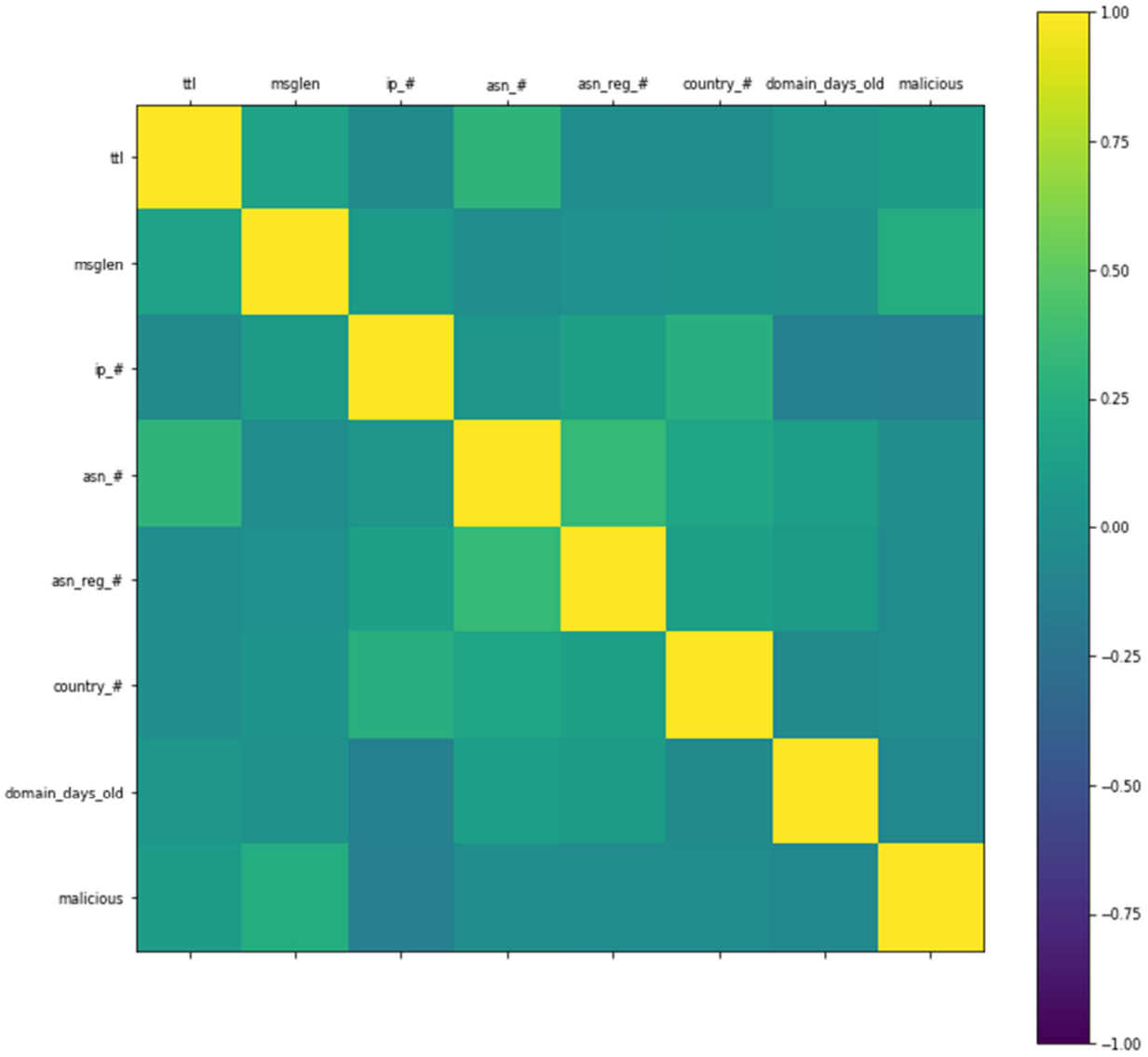
In de analyse fase heb ik de statistische samenhang tussen tweetallen van features onderzocht en tussen feature, ook wel voorspeller genoemd, en doelvariabele (malicious). Een correlatie is positief wanneer beide features in dezelfde richting bewegen. Wanneer features tegengesteld bewegen, is dit een negatieve correlatie. Wanneer features geen effect op elkaar hebben, is er sprake van een neutrale correlatie.

Om de samenhang tussen de verschillende numerieke features inzichtelijk te maken, wordt een correlatiematrix opgesteld, ook wel Pearson's r genoemd. Deze geeft de lineaire correlatie weer tussen twee features. Wanneer de standaard correlatiecoëfficiënt dicht tegen de één aan zit, is er een sterke positieve relatie tussen de twee features. Bijvoorbeeld, wanneer er meerdere landen zijn (country_count), is de kans groot dat er ook meerdere registrars bij betrokken zijn (asn_reg_count). Wanneer de standaard correlatiecoëfficient van deze twee features dicht tegen de min één aan zit, is er een sterke negatieve relatie. Als de standaard correlatiecoëfficient dicht bij de nul zit, betekent het dat er geen lineaire correlatie tussen bestaat.

In Figuur 27 en Figuur 28 zijn de uitkomsten van de correlatiematrix van respectievelijk DatasetI en DatasetII weergegeven. Voorspellers die geen enkele samenhang vertonen met de doelvariabele (waarde Pearson's r = 0), zijn features die mogelijk beter verwijderd kunnen worden. Uit beide grafieken is op te maken dat de voorspellers 'asn#' (asn_count) en 'asn_reg_#' (asn_reg_count) weinig samenhang vertonen. Voor het definitieve model zijn ze dus mogelijk niet relevant.
Wanneer features onderling sterk met elkaar gecorreleerd zijn, kan dit betekenen dat deze erg op elkaar lijken. Om het model eenvoudiger te maken en met minder kans op ruis, kan één van de features vervallen. In beide grafieken is te zien dat de correlatie tussen de features 'asn_reg_#' en 'asn_#' het grootst is.
De verschillen tussen de twee datasets komen ook naar voren in deze grafieken. Bij DatasetI is 'domain_days_old' de beste voorspeller, gevolgd door 'msglen' en bij DatasetII is dit 'msglen' gevolgd door 'ip_#'.



FIGUUR 27 CORRELATIE MATRIX FEATURES DATASETI



FIGUUR 28 CORRELATIE MATRIX FEATURES DATASETII

Met de informatie uit deze analysefase kan bij het trainen van het model, onderzocht worden of er verbeteringen mogelijk zijn door het verwijderen van sommige features. In het kader van tijd, heb ik deze stap niet gedaan en heb ik alle tien features gebruikt.

6.1.4 Data klaarmaken: feature engineering

In deze stap wordt de trainingdata geprepareerd voordat die wordt aangeboden aan een machine learning algoritme. Op basis van de eerder gemaakte analyse zijn de volgende bewerkingen uitgevoerd: null-waarden aanpassen, waarden van features schalen zodat ze gelijksoortig zijn en niet-numerieke features omzetten naar numerieke waarden. Vervolgens wordt een 'pipeline' gedefinieerd waarmee deze stappen geautomatiseerd kunnen worden uitgevoerd bij het trainen en testen van de diverse modellen.

Null-waarden

Met null-waarden kan op diverse manieren worden omgegaan. De betreffende rijen kunnen bijvoorbeeld uit de dataset worden verwijderd, de hele feature kan worden verwijderd of de null-waarde kan worden vervangen. In mijn dataset wil ik zowel alle features als alle rijen behouden omdat de dataset niet heel groot is.
Als het model in de praktijk wordt gebruikt zullen we ook null-waarden tegenkomen. De vervangers hiervan die we nu bepalen, moeten daarom onthouden worden zodat ze ook in de praktijk kunnen worden toegepast.

Van de tien features zijn er acht die null-waarden bevatten. Hiervan zijn er drie van het type object en vijf numeriek. Bij de object features veranderen we de null-waarden in 'empty'. Het niet aanwezig zijn van data kan namelijk een mogelijk teken zijn dat het domein malafide is. Voor de missende numerieke waarden, is gekozen om de mediaanwaarde van de betreffende feature te berekenen en deze waarde te gebruiken. Binnen Scikit-Learn heb ik gekozen om de functie 'SimpleImputer' hiervoor te gebruiken.

Object features omzetten
Machine learning algoritmen werken het beste met numerieke waarden. Features die een categorie als waarde hebben, moeten omgezet worden naar numerieke waarden. Binnen Scikit-Learn heb ik hiervoor van de functie 'OneHotEncoder' gebruikt gemaakt. Deze functie zet categorische features om in 'one-hot vectors'. Een one-hot vector is een 1xn matrix waarbij alleen een één komt te staan bij de categorie die van toepassing is.

Schalen van features
Het schalen van features is een van de belangrijkste onderdelen van het preparen van een dataset en wordt ook wel normalisatie genoemd. In de praktijk zullen de waarden van de onbewerkte features sterk uiteen lopen. Zo ook bij mijn datasets: de schaal van feature "ttl" loopt van nul tot meer dan drie miljoen en van "ip_count" van 1 tot en met 17. Bijna alle machine learning algoritmen kunnen hier slecht mee overweg. Wanneer een feature een veel groter bereik heeft dan andere features, zal deze in de berekening van het algoritme veel zwaarder wegen en daarmee een vertekend beeld geven.
Voor het schalen van features zijn diverse mogelijkheden. Twee veelgebruikte manieren zijn 'min-max scaling', ook wel normalisatie genoemd, en 'standardization'. Min-max scaling is de eenvoudigste methode. Waarden worden opnieuw berekend zodat de waarden liggen tussen nul en één. Dit wordt gedaan door eerst de minimum waarde op te zoeken en deze van alle waarden af te trekken. Vervolgens worden de waarden gedeeld door het verschil tussen de minimumwaarde en de maximumwaarde. Dit kan als volgt worden weergegeven, waarbij x een originele waarde is en x' de bijbehorende genormaliseerde waarde:

$$x' = \frac{\mathrm{x} - \min(\mathrm{x})}{\max(x) - \min(x)}$$

Bij standaardisatie worden de waarden gecentreerd rond het gemiddelde met een standaarddeviatie van de feature. Het gemiddelde van de feature wordt hiermee nul en de resterende verdeling heeft een standaarddeviatie van de feature. Dit kan als volgt worden weergegeven, waarbij μ het gemiddelde is van de feature waarden en σ de standaarddeviatie:

$$x' = \frac{\mathrm{x} - \mu}{\sigma}$$

Bij standaardisatie zijn de waarden niet beperkt tot een bepaald bereik. Standaardisatie is minder gevoelig voor uitschieters dan min-max scaling. Echter, min-max scaling past weer beter bij een dataset waarbij de features geen normaal verdeling hebben.
Voor beide methoden is iets te zeggen: mijn datasets bevatten uitschieters maar hebben grotendeels geen normaal verdeling. De voorkeur zou in dit geval gaan naar het toepassen van beide methoden en bekijken welke uiteindelijk tot de beste resultaten leidt.
Uit praktische overwegingen is gekozen voor standaardisatie waarbij ik binnen Scikit-Learn gekozen heb voor de functie 'Standardscaler'.

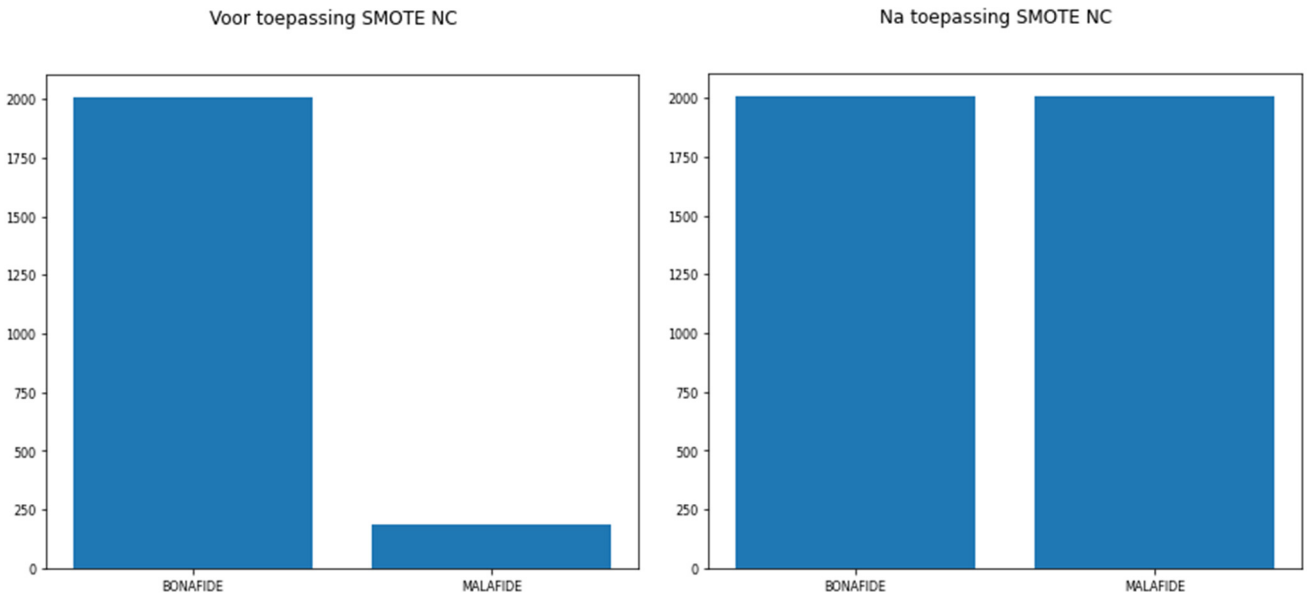
Pipeline
Binnen Scikit-Learn is het mogelijk om gebruik te maken van 'pipelines'. Dit houdt in dat alle bewerkingen in functies omschreven zijn en achter elkaar worden uitgevoerd. Hiermee heb ik de datapreparatie geautomatiseerd en kunnen deze stappen ook eenvoudig worden toegepast op de testset.

Na het uitvoeren van de hierboven omschreven stappen is de trainingset klaar om gebruikt te worden door een te kiezen algoritme.

6.1.5 Dataset balanceren

In DatasetI zijn 2867 malafide domeinnamen aanwezig en 2511 bonafide domeinnamen. De dataset is in balans. DatasetII bevat 233 malafide domeinnamen en 2511 bonafide domeinnamen. DatasetII is door 'oversampling' van de malafide klasse gebalanceerd. Oversampling houdt in dat het aantal datapunten van de minderheidsklasse worden vermeerderd. Oversampling kan alleen worden uitgevoerd wanneer er geen null-waarden aanwezig zijn. Vandaar dat deze stap wordt uitgevoerd nadat DatasetII is geprepareerd. Ook is oversampling alleen toegepast op de trainingset. Zo blijft de testset ten allen tijde vrij van bewerkingen om in een later stadium mijn modellen hiermee te valideren.

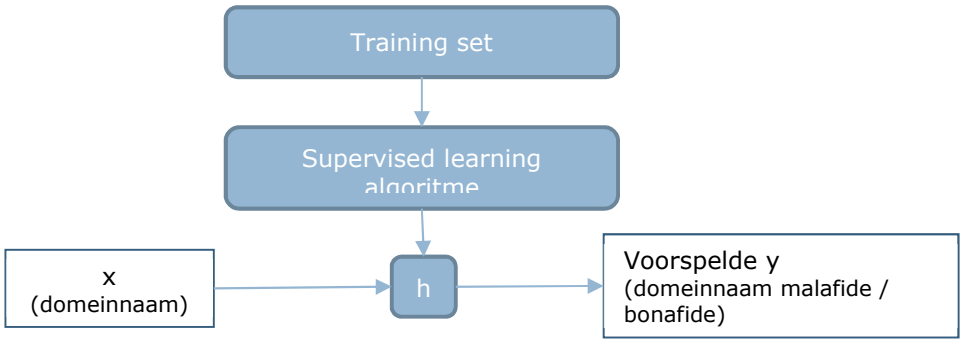
Een van de meest gebruikte oversampling technieken is Synthetic Minority Oversampling Technique (SMOTE) (Viloria, Lezama, & Mercado-Caruzo, 2020), (Amin, et al., 2016). SMOTE werkt op basis van het maken van nieuwe synthetische datapunten op basis van voorbeelden uit de minderheidsklasse. Voor DatasetII is de variant SMOTE NC gebruikt omdat deze niet alleen nieuwe datapunten kan creëren van numerieke features maar ook van categorische features. SMOTE maakt geen kopieën van aanwezige datapunten maar creëert nieuwe datapunten op basis van kenmerken van bestaande datapunten en combineert deze met kenmerken van de buren.
Nadat DatasetII is gesplitst in een trainingset en testset, bevat de trainingset 2006 bonafide domeinnamen en 189 malafide domeinnamen. Na het toepassen van SMOTE NC bevat deze set 2006 bonafide en 2006 malafide domeinnamen, zie Figuur 29.



FIGUUR 29 AANTAL BONAFIDE EN MALAFIDE DOMEINEN VOOR / NA TOEPASSING SMOTE NC

6.2 Concept machine learning model

Bij het opzetten van een concept model is één van de belangrijkste onderdelen het kiezen van het juiste algoritme en dit algoritme verder finetunen. Het model kunnen we als volgt formeel omschrijven: met een gegeven trainingset een functie $h: X \rightarrow Y$ te leren zodat h(x) een goede voorspeller is voor de corresponderende waarde van y (Ng, -). Deze functie wordt ook wel hypothese genoemd. Schematisch ziet het er als volgt uit (Figuur 30):



FIGUUR 30 MACHINE LEARNING MODEL REPRESENTATIE (ANDREW NG, MACHINE LEARNING)

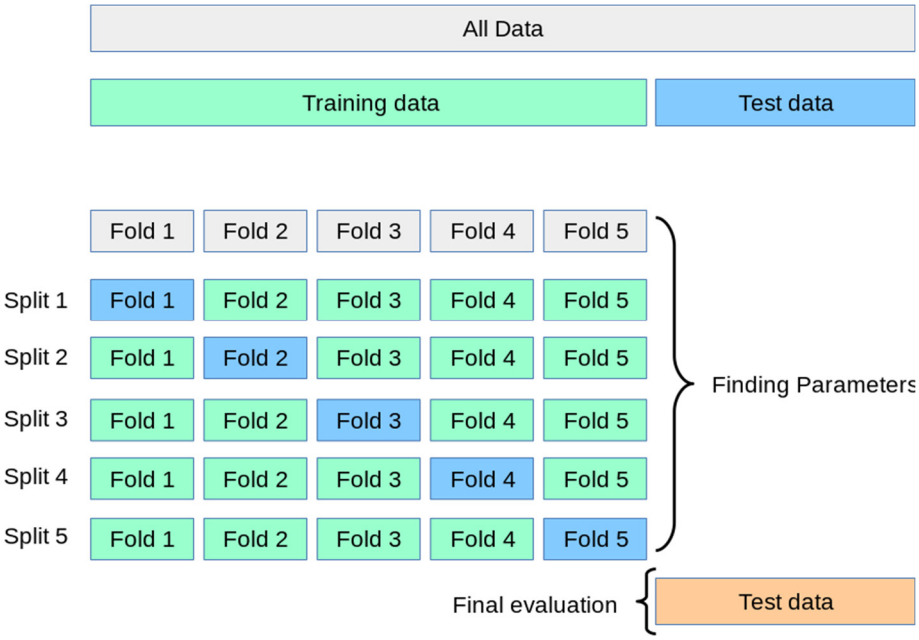
Om te komen tot een definitief model wordt eerst een aantal verschillende machine learning algoritmen op hoofdlijnen doorgerekend met de trainingset. Deze worden nog niet gefinetuned. Op deze manier is snel een indruk verkregen hoe ze presteren. Hierbij moet opgemerkt worden dat het kan voorkomen dat een bepaald algoritme minder goed werkt met basis instellingen en de test dus mogelijk een vertekend beeld geeft.
Als input voor de algoritmen gebruik ik de trainingsets van DatasetI en DatasetII. Dit betekent dat ik elk algoritme ook tweemaal heb doorgerekend.

Mijn modellen heb ik beoordeeld met de volgende indicatoren: Confusion matrix, Precision, Recall en F1 en AUC. Deze evaluatiemethoden zijn toegelicht in paragraaf 4.6.2.

6.2.1 Diverse algoritmes doorrekenen

Voor mijn modelbouw heb ik de volgende zes machine learning algoritmen gekozen, waarbij tussen haakjes de betreffende functie binnen Scikit-Learn is aangegeven: Support Vector Machine (SGDClassifier), Decision tree (DecisionTreeClassifier), Random Forests (RandomForestClassifier), Naïve Bayes (GaussanNB), k-Nearest Neighbors (KNeigborsClassifier) en Logistic Regression (LogisticRegression). Dit zijn veel gebruikte algoritmen binnen het machine learning veld. De algoritmen zijn in paragraaf 334.6.1 nader toegelicht.

De zes algoritmen zijn getraind met de twee trainingsets en geëvalueerd door middel van 'k-fold cross validatie'. Dit houdt in dat elke trainingset is opgedeeld in *k* folds, ook wel subset genoemd. Voor het trainen wordt *k*-1 folds gebruikt en voor het valideren de overgebleven subset. Telkens wordt een ander deel gebruikt om te valideren. Bijvoorbeeld als k = 5, dan wordt elke trainingset in 5 delen opgesplitst. Vijf keer worden dan vier folds gebruikt om te trainen en één fold om te valideren, waarbij bij elke ronde een ander deel gebruikt wordt voor de validatie, zie Figuur 31.



FIGUUR 31 SCHEMATISCHE WEERGAVE CROSSVALIDATIE (SCIKIT LEARN)

Vervolgens zijn de performance indicatoren confusion matrix, Precision, Recall, F1 en AUC berekend.

In Tabel 9 staan de uitkomsten van de evaluatie. Per perfomance indicator is in blauw aangegeven wat de beste waarde is bij gebruik van DatasetI en in lichtblauw bij gebruik van DatasetII. Uit de tabel is op te maken dat het algoritme Random Forest het beste presteert. Hoewel de Recall-waarde niet het hoogste is, is deze nog zeer goed te noemen: 0,966 (DatasetI) en 0,979 (DatasetII). Dit houdt in dat respectievelijk 96.6% en 97.9% van alle malafide domeinnamen herkend is als malafide.

TABEL 9 OVERZICHT PERFORMANCE PER ALGORITME

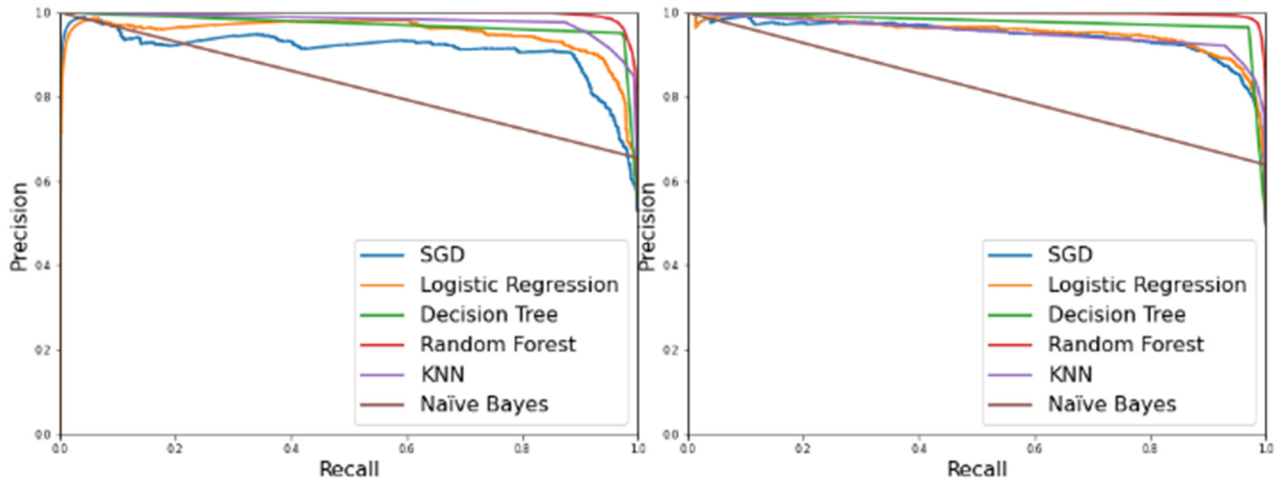
		Confusion matrix				Precision	Recall	F1	AUC
Support Vector Machine	DatasetI			Voorspelling		0,868	0,902	0,885	0,921
				Negatief	Positief				
		actuele waarde	Negatief	1704	313				
			Positief	223	2062				
	DatasetII			Voorspelling		0,874	0,929	0,901	0,957
				Negatief	Positief				
		actuele waarde	Negatief	1738	268				
			Positief	142	1864				
Logistic Regression	DatasetI			Voorspelling		0,908	0,912	0,910	0,957
				Negatief	Positief				
		actuele waarde	Negatief	1806	211				
			Positief	202	2083				
	DatasetII			Voorspelling		0,893	0,922	0,907	0,962
				Negatief	Positief				
		actuele waarde	Negatief	1784	222				
			Positief	156	1850				
Decision Tree	DatasetI			Voorspelling		0,951	0,973	0,962	0,960
				Negatief	Positief				
		actuele waarde	Negatief	1903	114				
			Positief	61	2224				
	DatasetII			Voorspelling		0,965	0,971	0,968	0,968
				Negatief	Positief				
		actuele waarde	Negatief	1953	71				
			Positief	59	1947				
Random Forest	DatasetI			Voorspelling		0,976	0,966	0,971	0,995
				Negatief	Positief				
		actuele waarde	Negatief	1963	54				
			Positief	78	2207				
	DatasetII			Voorspelling		0,983	0,979	0,981	0,997
				Negatief	Positief				
		actuele waarde	Negatief	1972	34				
			Positief	43	1963				
k-Nearest Neighbors	DatasetI			Voorspelling		0,915	0,951	0,931	0,976
				Negatief	Positief				
		actuele waarde	Negatief	1815	202				
			Positief	111	2174				
	DatasetII			Voorspelling		0,837	0,983	0,904	0,951
				Negatief	Positief				
		actuele waarde	Negatief	1621	385				
			Positief	34	1972				
Naïve Bayes	DatasetI			Voorspelling		0,657	0,996	0,792	0,704
				Negatief	Positief				
		actuele waarde	Negatief	830	1187				
			Positief	8	2277				
	DatasetII			Voorspelling		0,640	0,999	0,780	0,719
				Negatief	Positief				
		actuele waarde	Negatief	880	1126				
			Positief	3	2003				

Uit de bovenstaande tabel is af te lezen dat de modellen verschillend presteren op basis van de gebruikte Datasets. De verschillen in prestaties is bij de performance indicator Precision het

grootst en varieert tussen de 0,7% en 8,5% in het voordeel van DatasetI. Bij de andere performance indicatoren zijn de verschillen kleiner. Op basis van deze uitkomsten kan geen sterke voorkeur voor een dataset uitgesproken worden.

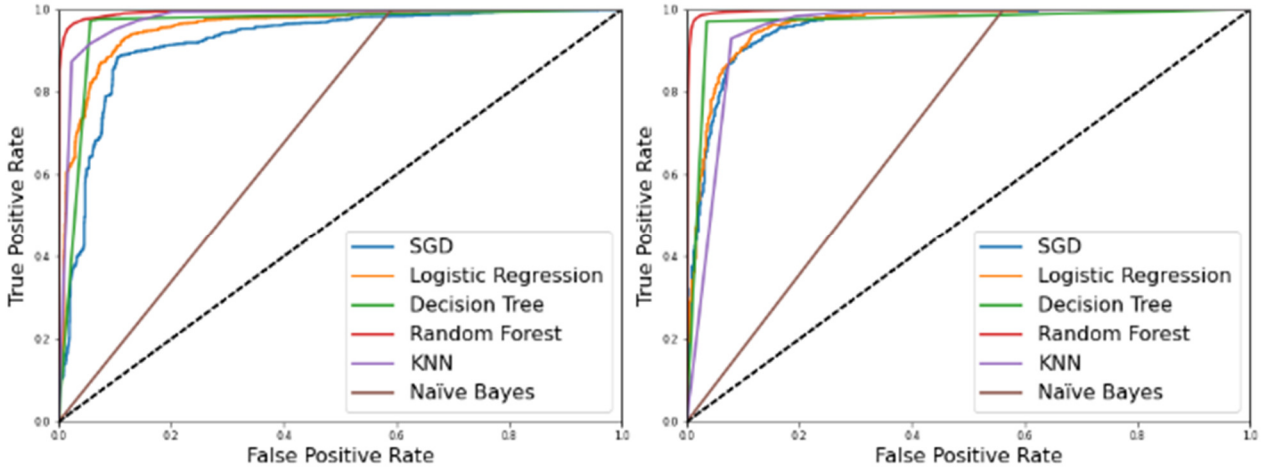
6.2.2 Algoritme selectie

Naast het uitlezen van de performance indicatoren, is het aflezen van de verhouding tussen Precision en Recall een veel gebruikt hulpmiddel om classificatie algoritmen te beoordelen. Deze verhouding geeft weer hoe goed een algoritme de juiste voorspellingen doet in verhouding tot het totaal aantal juiste voorspellingen van de positieve klasse. In mijn geval is de positieve klasse: een malafide domeinnaam.
In Figuur 32 is deze verhouding per algoritme en per dataset weergegeven. Hoe dichter de betreffende lijn bij de rechter bovenhoek van de grafiek komt, des te beter scoort het algoritme. Uit beide grafieken komt naar voren dat Random Forest het best presterende algoritme is.



FIGUUR 32 PRECISION VS RECALL VAN DE DIVERSE ALGORITMEN- DATASETI (LINKS) EN DATASETII (RECHTS)

Voor de beoordeling van binaire classificatie algoritmen wordt ook vaak de 'receiver operating characterisc' curve (ROC) gebruikt (zie Figuur 33). Deze is vergelijkbaar met de voorgaande grafiek, maar in plaats van het afzetten van de nauwkeurigheid tegenover het aantal juist gelabelde positieve uitkomsten, wordt nu het totaal aantal terechte positief gelabelde uitkomsten gezet tegenover het totaal aantal ten onrechte positief gelabelde uitkomsten gezet. Hoe dichter de betreffende lijn bij de linker bovenhoek van de grafiek komt, des te beter functioneert het algoritme.



FIGUUR 33 ROC-CURVE VAN DE DIVERSE ALGORITMEN- DATASETI (LINKS) EN DATASETII (RECHTS)

Van de zes algoritmen heb ik uiteindelijk 3 algoritmen gekozen om nader te onderzoeken. Eén daarvan is Random Forest, volgens de grafieken het best presterende algoritme. Ondanks dat Decision Tree het op één na best presterende algoritme is, heb ik niet voor dit algoritme gekozen. Vanwege de mogelijke instabiliteit alsook de vergelijkbaarheid met Random Forest, heb ik besloten deze niet te kiezen.
De prestaties van KNN, Logistic Regression en SVM (SGD in de grafieken) zijn sterk vergelijkbaar waarbij het lijkt dat KNN iets beter presteert bij DatasetI en iets minder bij DatasetII. In de volgende stap heb ik er voor gekozen om naast KNN ook het algoritme SVM nader te onderzoeken. KNN omdat deze ook zeer goed presteert bij DatasetI en SVM omdat dit algoritme bekend staat goed te kunnen presteren bij ongebalanceerde data. Ik heb hierbij gekozen voor SGDClassifer binnen Scikit Learn. Stochistic Gradient Descent is een optimalisatietechniek dat wordt gebruikt in combinatie met verschillende leertechnieken. Wanneer gebruik gemaakt wordt van SGDClassifier waarbij de hyperparameter 'loss' op 'hinge' wordt gezet, werkt de SGDClassifier als een linear SVM algoritme. SGDClassifier is ten opzichte van een linear SVM eenvoudig te implementeren en is een efficiënt algoritme.

6.3 Modellen afstellen en trainen

In deze stap zijn de hyperparameters, de parameters die specifiek algoritme-afhankelijk zijn, zodanig ingesteld dat een algoritme het meest optimaal presteert bij mijn trainingsets. Dit wordt ook wel hyperparameteroptimalisatie genoemd. Vervolgens wordt het model hiermee getraind en worden de trainingsresultaten beoordeeld of de uitgevoerde stappen tot verbeteringen hebben geleid.
Binnen Scikit-Learn kan daarvoor de functie Randomized Search en Grid Search worden gebruikt. Met Grid Search kunnen de hyperparameters worden opgegeven waarmee geëxperimenteerd moet worden, alsook welke waarden hierbij gebruikt moeten worden. Grid Search functie probeert vervolgens alle mogelijke combinaties van hyperparameters en geeft de best presterende combinatie voor het algoritme bij de betreffende dataset retour.
Wanneer er veel hyperparameters zijn, gaat de voorkeur uit naar Randomized Search. De werking is hetzelfde alleen wordt nu een opgegeven aantal willekeurige combinaties van waarden voor elke hyperparameter geprobeerd in plaats van alle mogelijke combinaties.

6.3.1 Random Forest

Bij het RandomForest algoritme heb ik eerst Randomized Search toegepast en daarna Grid Search. De gevonden hyperparameter instellingen zijn weergegeven in Tabel 10.

TABEL 10 OPTIMALE INSTELLINGEN HYPERPARAMETERS RANDOMFORESTCLASSIFIER BIJ DE TWEE DATASETS

Hyperparameterselectie Random Forest				
	DatasetI		DatasetII	
Parameter	Randomized	Grid	Randomized	Grid
Bootstrap	False	False	False	False
Max_deph	70	None	90	90
Max_features	Sqrt	Sqrt	Sqrt	Sqrt
Min_samples_leaf	1	1	1	1
Min_samples_split	5	2	5	4
N_estimators	1600	100	800	800

Na uitvoering van Randomized Search en Grid Search is het algoritme opnieuw geëvalueerd. Ik heb daarbij gebruik gemaakt van crossvalidatie om te beoordelen of de performance is verbeterd, waarbij ik de training data heb opgesplitst in 5 folds (k = 5). De resultaten hiervan zijn weergegeven in Tabel 11.

TABEL 11 UITKOMSTEN VERBETERINGEN NA OPTIMALISATIE RANDOMFORESTCLASSIFIER

Random Forest									
		Confusion matrix				Precision	Recall	F1	AUC
Vóór hyperparameter optimalisatie	DatasetI			Voorspelling		0,976	0,966	0,971	0,995
				Negatief	Positief				
		actuele waarde	Negatief	1963	54				
			Positief	78	2207				
	DatasetII			Voorspelling		0,983	0,979	0,981	0,997
				Negatief	Positief				
		actuele waarde	Negatief	1972	34				
			Positief	43	1963				
Optimalisatie stap1: Randomized Search	DatasetI			Voorspelling		0,976	0,975	0,975	0,974
				Negatief	Positief				
		actuele waarde	Negatief	1962	55				
			Positief	57	2228				
	DatasetII			Voorspelling		0,982	0,989	0,986	0,986
				Negatief	Positief				
		actuele waarde	Negatief	1970	36				
			Positief	22	1984				
Optimalisatie stap2: Grid Search	DatasetI			Voorspelling		0,974	0,968	0,971	0,969
				Negatief	Positief				
		actuele waarde	Negatief	1957	60				
			Positief	72	2213				
	DatasetII			Voorspelling		0,978	0,986	0,982	0,982
				Negatief	Positief				
		actuele waarde	Negatief	1962	44				
			Positief	29	1977				

Uit de tabel blijkt dat verbetering bij beide datasets na het uitvoeren van Randomized Search minimaal is en de AUC score enigszins verminderd.

Bij DatasetI zijn de uitkomsten als volgt:

Precision: 0.0%
Recall: 0.9%
F1: 0.4%
AUC: -2.1%

Bij DatasetII zijn de uitkomsten als volgt:

Precision: -0.1%
Recall: 1.0%
F1: 0.5%
AUC: -1.1%

De performance van alle 4 indicatoren is na het uitvoeren van Grid Search verminderd voor beide datasets. Dit in tegenstelling tot de hyperparameters zoals die gevonden zijn na de Randomized Search. Deze zullen dan ook worden gebruikt bij het definitieve model.

6.3.2 K-Nearest Neighbors

Het aantal parameters is bij het k-Nearest Neighbors algoritme zeer beperkt, daarom kan worden volstaan met alleen het uitvoeren van Grid Search. Er is gezocht naar de optimale waarde van hyperparameter 'k', het aantal buren waarnaar wordt gekeken, en naar 'weights', de mate waarin elke buur wordt meegewogen. Voor 'k' is gestart met waarden van 1 tot 23. En voor 'weights' de waarden 'uniform' en 'distance'. Uit de zoektocht is naar voren gekomen dat de optimale waarden voor DatasetI als volgt zijn: k=1 en weights = uniform, en voor DatasetII k=7 en weights = distance.

Na de Grid Search wordt het algoritme opnieuw geëvalueerd met de trainingsets, door middel van crossvalidatie om te beoordelen of de performance is verbeterd, waarbij ik de training data heb opgesplitst in 5 folds (k = 5). De resultaten hiervan zijn weergegeven in Tabel 12.

TABEL 12 UITKOMSTEN VERBETERINGEN NA OPTIMALISATIE KNEIGBORSCLASSIFIER

k-Nearest Neighbors								
		Confusion matrix			Precision	Recall	F1	AUC
Voor hyperparameter optimalisatie	DatasetI		Voorspelling: Negatief	Voorspelling: Positief				
		actuele waarde: Negatief	1815	202				
		actuele waarde: Positief	111	2174	0,915	0,951	0,931	0,976
	DatasetII		Voorspelling: Negatief	Voorspelling: Positief				
		actuele waarde: Negatief	1621	385				
		actuele waarde: Positief	34	1972	0,837	0,983	0,904	0,951
Na hyperparameter optimalisatie (Grid Search)	DatasetI		Voorspelling: Negatief	Voorspelling: Positief				
		actuele waarde: Negatief	1886	131				
		actuele waarde: Positief	70	2215	0,974	0,968	0,971	0,952
	DatasetII		Voorspelling: Negatief	Voorspelling: Positief				
		actuele waarde: Negatief	1803	203				
		actuele waarde: Positief	39	1967	0,906	0,981	0,942	0,940

Hieruit blijkt dat de verbetering bij beide datasets na het uitvoeren van Grid Search voor de indicatoren Precision en F1 noemenswaardig is en de AUC score enigszins verminderd is.

Bij DatasetI zijn de uitkomsten als volgt:
- Precision: 5.9%
- Recall: 1.7%
- F1: 4.0%
- AUC: -2.4%

Bij DatasetII zijn de uitkomsten als volgt:
- Precision: 6.9%
- Recall: -0.2%
- F1: 3.8%
- AUC: -1.1%

6.3.3 SVM

Ook bij het Support Vector Machine algoritme SGDClassifier is gekozen voor alleen het uitvoeren van een GridSearch. In Tabel 13 is de input weergegeven voor de Gridsearch en de uitkomst van de meest optimale instellingen voor de hyperparameters. De eerste parameter 'Loss' is een vast gegeven om het algoritme als een linear SVM te laten werken.

TABEL 13 OPTIMALE INSTELLINGEN HYPERPARAMETERS SGDCLASSIFIER BIJ DE TWEE DATASETS

Hyperparameterselectie SVM : SGDClassifier			
Parameter	Input	Output DatasetI	Output DatasetII
Loss	hinge	hinge	hinge
Alpha	0.1, 0.01, 0.001, 0.0001	0.0001	0.0001
Penalty	l2, l1, elasticnet, none	l1	none

Na de Grid Search wordt het algoritme opnieuw geëvalueerd met de trainingsets, door middel van crossvalidatie om te beoordelen of de performance is verbeterd, waarbij ik de training data heb opgesplitst in 5 folds (k = 5). De resultaten hiervan zijn weergegeven in Tabel 14.

TABEL 14 UITKOMSTEN NA OPTIMALISATIE SUPPORT VECTOR MACHINE: SGDCLASSIFIER

Support Vector Machine								
		Confusion matrix			Precision	Recall	F1	AUC
Voor hyperparameter optimalisatie	DatasetI		Voorspelling: Negatief	Voorspelling: Positief				
		actuele waarde: Negatief	1704	313				
		actuele waarde: Positief	223	2062	0,868	0,902	0,885	0,921
	DatasetII		Voorspelling: Negatief	Voorspelling: Positief				
		actuele waarde: Negatief	1738	268				
		actuele waarde: Positief	142	1864	0,874	0,929	0,901	0,957
Na hyperparameter optimalisatie (Grid Search)	DatasetI		Voorspelling: Negatief	Voorspelling: Positief				
		actuele waarde: Negatief	1814	203				
		actuele waarde: Positief	158	2127	0,913	0,931	0,923	0,915
	DatasetII		Voorspelling: Negatief	Voorspelling: Positief				
		actuele waarde: Negatief	1856	150				
		actuele waarde: Positief	94	1912	0,927	0,953	0,940	0,939

Hieruit blijkt dat de verbetering bij beide datasets na het uitvoeren van Grid Search voor de performance indicatoren Precision en F1 het grootste zijn en de AUC score bij beide datasets enigszins verminderd is.

Bij DatasetI zijn de uitkomsten als volgt:

- Precision: 4.5%
- Recall: 2.9%
- F1: 3.8%
- AUC: -0.6%

Bij DatasetII zijn de uitkomsten als volgt:

- Precision: 5.3%
- Recall: 2.4%
- F1: 3.9%
- AUC: -1.8%

6.4 Test resultaten

Nadat het model getraind is, wordt het model getest met de testset waarbij eerst de testset op dezelfde wijze wordt geprepareerd als de training set (zie paragraaf 6.1.4).
De testresultaten laten zien dat het model Random Forest de beste resultaten oplevert bij gebruik van DatasetI. Ook de andere twee modellen laten een acceptabel resultaat zien.
Hoewel de resultaten van de verschillende modellen bij gebruik van DatasetII veelbelovend leken, zijn de testresultaten hiervan slecht te noemen. De performance indicator Recall laat zien dat de verschillende modellen bij DatasetII zeer slecht in staat zijn om de malafide klasse te herkennen. De modellen hebben de neiging om alle domeinnamen als negatief (bonafide) aan te merken. Mogelijke oorzaak is dat de verhouding tussen malafide (klasse 1) en bonafide (klasse 0) domeinen in de testset van DatasetII scheef gebalanceerd is in verhouding tot de testset van DatasetI. Namelijk 44 malafide domeinnamen en 505 bonafide domeinnamen versus 582 malafide domeinnamen en 494 bonafide domeinnamen.
In Tabel 15 zijn de testresultaten weergegeven.

TABEL 15 TESTRESULTATEN RANDOMFORESTCLASSIFIER, KNEIGBORSCLASSIFIER, SGDCLASSIFIER

		Confusion matrix				Precision	Recall	F1	AUC
Random Forest	DatasetI			Voorspelling		0,947	0,948	0,948	0,943
				Negatief	Positief				
		actuele waarde	Negatief	463	31				
			Positief	30	552				
	DatasetII			Voorspelling		0,786	0,250	0,379	0,622
				Negatief	Positief				
		actuele waarde	Negatief	502	3				
			Positief	33	11				
k-Nearest Neighbors	DatasetI			Voorspelling		0,909	0,947	0,928	0,918
				Negatief	Positief				
		actuele waarde	Negatief	439	55				
			Positief	31	551				
	DatasetII			Voorspelling		0,636	0,159	0,255	0,576
				Negatief	Positief				
		actuele waarde	Negatief	501	4				
			Positief	37	7				
Support Vector Machine	DatasetI			Voorspelling		0,897	0,930	0,913	0,902
				Negatief	Positief				
		actuele waarde	Negatief	432	62				
			Positief	41	541				
	DatasetII			Voorspelling		0,326	0,318	0,322	0,630
				Negatief	Positief				
		actuele waarde	Negatief	476	29				
			Positief	30	14				

Omdat in de testsituatie de machine learning modellen gebaseerd op DatasetII allemaal zeer slecht presteerden, is een extra test uitgevoerd. De getrainde modellen zijn nogmaals getest maar nu met de andere dataset.
De uitkomsten hiervan zijn weergegeven in Tabel 16.

TABEL 16 TESTRESULTATEN MET DATASET WAARMEE NIET GETRAIND IS

			Confusion matrix				Precision	Recall	F1	AUC
Random Forest	Getraind met: DatasetI	Getest met: DatasetII			Voorspelling					
					Negatief	Positief				
			actuele waarde	Negatief	2471	13				
				Positief	114	119	0,902	0,511	0,652	0,753
	Getraind met: DatasetII	Getest met: DatasetI			Voorspelling					
					Negatief	Positief				
			actuele waarde	Negatief	2428	62				
				Positief	72	2763	0,978	0,975	0,976	0,975
k-Nearest Neighbors	Getraind met: DatasetI	Getest met: DatasetII			Voorspelling					
					Negatief	Positief				
			actuele waarde	Negatief	2432	52				
				Positief	93	140	0,729	0,601	0,659	0,790
	Getraind met: DatasetII	Getest met: DatasetI			Voorspelling					
					Negatief	Positief				
			actuele waarde	Negatief	2283	207				
				Positief	75	2760	0,930	0,974	0,951	0,945
Support Vector Machine	Getraind met: DatasetI	Getest met: DatasetII			Voorspelling					
					Negatief	Positief				
			actuele waarde	Negatief	2443	51				
				Positief	145	88	0,633	0,378	0,473	0,679
	Getraind met: DatasetII	Getest met: DatasetI			Voorspelling					
					Negatief	Positief				
			actuele waarde	Negatief	2246	244				
				Positief	166	2669	0,916	0,941	0,929	0,922

Nu zien we tegenovergestelde resultaten: modellen die getraind zijn met DatasetII maar getest worden met DatasetI scoren goed tot zeer goed. Ook nu is Random Forest model het best presterende model.

Alle modellen presteren rond de drie procent beter in vergelijking met wanneer alleen getraind en getest wordt met DatasetI. Daarentegen presteren de modellen die getraind zijn met DatasetI en getest zijn met DatasetII beduidend slechter. Echter, over de gehele linie gezien, is dit toch minder slecht dan wanneer de modellen alleen getraind en getest worden met DatasetII. Dit bevestigt mijn vermoeden dat een algoritme niet goed presteert wanneer de testset ongebalanceerd is. Dit ondanks het feit dat de trainingset vooraf is gebalanceerd. De testset is een weergave van de praktijksituatie. Of het model goed in de praktijk gaat werken is nog maar zeer de vraag.

6.5 Vergelijken met andere detectiemethoden

Het vergelijken van mijn testresultaten met andere DNS gebaseerde detectiemethoden is lastig. Eén van de redenen is dat detectiemethoden verschillende aandachtsgebieden kunnen hebben. Bijvoorbeeld het herkennen van malafide domeinnamen of het herkennen van afwijkingen in DNS verkeer. Ook de dataset waarop getraind wordt heeft invloed, zo zal een dataset welke gebaseerd is op ISP verkeer andere kenmerken bevatten dan een dataset die gebaseerd is op een bedrijfsnetwerk. Datasets kunnen uiteraard ook hele verschillende voorbeelden van type botnets bevatten. Echter, al zouden bovenstaande verschillen niet bestaan, dan is het nog steeds lastig om een vergelijking te maken. De detectiemethoden die ik in mijn literatuuronderzoek ben tegen gekomen, bevatten niet allemaal performance indicatoren. Ook als een onderzoek deze wel bevat dan is het helaas lang niet altijd duidelijk waar die op gebaseerd zijn: trainingset, testset, of hoe de dataset eruit ziet. Tot slot worden er geen uniforme indicatoren gebruikt.

Als een van de weinigen, beschrijven Hoang & Nuyen welke performance indicatoren ze gebruiken en hoe deze berekend worden (Hoang & Nguyen, 2018). Hun performance indicator 'True Positive Rate' komt overeen met de, in mijn onderzoek beschreven, indicator 'Recall' en 'Positive Predictive Value' overeen met 'Precision'. Hun performance indicator 'F1' komt qua naamgeving overeen met de door mij gebruikte indicator maar de manier waarop die berekend wordt, verschilt. Ook bij de door hen gebruikte performance indicator 'False Positive Rate' is er geen overeenkomst met de in deze scriptie beschreven indicatoren.
Deze laatste wordt onder andere, samen met de performance indicator 'Detection rate', in diverse andere onderzoeken waaronder Xu et al., gebruikt. (Xu, Shen, & Du, 2019). Zij definiëren deze laatste als volgt: $Detection\ rate = \frac{True\ positive\ (TP)}{True\ positive\ (TP) + False\ Negative\ (FN)}$.
Hoewel het dus toch enigszins appels met peren vergelijken blijft, is in
Tabel 17 de uitkomst van mijn onderzoek vergeleken met twee andere onderzoeken.
Hierbij zijn de prestatie indicatoren weliswaar anders genoemd maar wel op dezelfde manier berekend. De resultaten uit het eigen onderzoek komen redelijk overeen met de uitkomsten van deze twee onderzoeken.
Hoang & Nguyen constateren in hun onderzoek een verschil in prestaties als de gebruikte datasets scheef gebalanceerd zijn (Hoang & Nguyen, 2018). Ze concluderen dat voor de beste prestaties een gebalanceerde dataset nodig is. Het best presterende model is het model waar gebruik gemaakt wordt van het Random Forest (RF) algoritme.
Dit komt overeen met mijn bevindingen.
Onderzoeken waar een hogere detection rate wordt gehaald zijn onder andere de volgende: Perdisci et al. (99,4%), Choi & Lee (97.2%) en Bilge et al. (99.5%) (Perdisci, Corona, Dagon, & Lee, 2009), (Choi & Lee, 2012), (Bilge, Kirda, Kruegel, & Balduzzi, 2011).

TABEL 17 VERGELIJKING UITKOMST EIGEN ONDERZOEK MET ANDERE ONDERZOEKEN

			Accuracy	False Positive rate (FP / (FP+TN)	Precision / PPV	Detection rate / True Positive rate / Recall
Best presterende model eigen onderzoek	RF	DatasetI trainen en testen:582 malafide domeinnamen en 494 bonafide domeinnamen	94,3%	6,1%	94,7%	94,8%
	RF	DatasetII trainen en DatasetI testen: 2835 malafide domeinnamen en 2490 bonafide domeinnamen	97,5%	2,1%	97,8%	97,5%
Xu et al	Bigram	verhouding klasse malafide en bonafide niet duidelijk	92,47%	6,63%	93,76%	91,64%
	Trigram	verhouding klasse malafide en bonafide niet duidelijk	96,2%	2,32%	97,75%	94,86%
Hoang & Nguyen	kNN	Training set 1: 10.000 malafide domeinnamen en 10.000 bonafide domeinnamen	90,2%	10,7%	89,5%	91,0%
	C4.5		90,1%	11,1%	89,1%	91,2%
	RF		90,8%	9,3%	90,7%	91,0%
	NB		85,9%	18,3%	83,1%	90,2%
	kNN	Training set 1: 15.000 malafide domeinnamen en 5.000 bonafide domeinnamen	88,1%	20,1%	82,7%	96,4%
	C4.5		87,6%	22,1%	81,5%	97,3%
	RF		89,2%	18,1%	84,2%	96,6%
	NB		85,8%	18,8%	82,8%	90,5%
	kNN	Training set 1: 5.000 malafide domeinnamen en 15.000 bonafide domeinnamen	87,7%	5,0%	94,1%	80,4%
	C4.5		87,6%	5,7%	93,4%	80,9%
	RF		88,1%	4,8%	94,4%	80,9%
	NB		86,0%	17,1%	83,9%	89,1%

6.6 Conclusie

De resultaten van het definitieve model zijn goed te noemen, waarbij er relatief weinig features (tien in totaal) gebruikt worden. Het toepassen van algoritme Random Forest in combinatie met DatasetII voor training en DatasetI voor testen (gebalanceerde testset) geeft de beste resultaten:

- Precision score (aantal juiste voorspellingen) van 97.8%,
- AUC score (vermogen om valse classificatie te vermeiden) van 97.5%,
- Recall score (de effectiviteit van een classificatie algoritme om de positieve labels te identificeren) van 97.5%.

Wanneer de onderzochte modellen, los van het gekozen algoritme, een ongebalanceerde testset krijgen, gaan de resultaten drastisch naar beneden en zijn de uitkomsten slecht te noemen. Kijken naar een aantal evaluatiescores die van belang zijn, dan zien we dat in het beste geval de Precision score rond de 90% ligt en de AUC score rond de 75%. Dat lijkt nog redelijk positief, echter, de Recall score ligt rond de 50%. Dat betekent dat elk malafide domein maar 50% kans heeft om ontdekt te worden. Juist het ontdekken van malafide domeinen is van belang om botnetbesmettingen te detecteren.

Zoals in hoofdstuk 4 reeds is geconcludeerd, is het vergelijken met andere DNS detectiemethoden erg lastig. Vanwege het feit dat randvoorwaarden anders ingevuld zijn en performance indicatoren verschillen van elkaar. Ook zijn lang niet bij alle onderzoeken de performance indicatoren te achterhalen. Zo worden niet altijd performance indicatoren gegeven, is het onduidelijk hoe deze berekend zijn of ontbreekt een beschrijving van de testset. Ook worden wel de resultaten gegeven van de performance indicatoren gebaseerd op de trainingset zoals bijvoorbeeld bij Hoang & Nguyen (Hoang & Nguyen, 2018). Daar waar toch een vergelijking gemaakt kon worden, blijkt niet veel verschillen.
Hoang & Nguyen concluderen op basis van hun uitkomsten dat een scheef gebalanceerde dataset negatieve invloed heeft op de resultaten. Dit komt ook in mijn onderzoek naar voren. In de overige onderzoeken wordt hier weinig tot geen aandacht aan besteed. Dit is opvallend want in de praktijk heeft men bij botnetdetectie hier vooral mee te maken: het overgrote deel is bonafide verkeer, slechts een klein deel is malafide. Opvallend detail in het onderzoek van Hoang & Nguyen is dat er verschillende trainingsets en een testset beschreven zijn, maar dat de uitkomsten van de verschillende performance indicatoren alleen beschreven zijn op basis van de trainingsets.
Tot slot zijn er wel onderzoeken, waaronder bijvoorbeeld Wang et al., die gebaseerd zijn op data uit de praktijk, maar ik heb geen onderzoeken kunnen vinden waar theoretische detectiemethoden daadwerkelijk zijn toegepast op praktijk situaties (Wang, Lin, Cheng, & Chen, 2017). In de Github omgeving heb ik gezocht naar de broncode van beschreven botnetdetectie methoden, zoals DBod, Exposure, DFBotKiller, BotGAD en FastFlux Hunter. Ik heb geen zoekresultaten terug gekregen die hier naar verwijzen. Dit vind ik een opvallend en niet verwacht resultaat. Het zou mijns inziens interessant zijn om te onderzoeken hoe de verschillende detectiemethoden presteren in een praktijksituatie. Niet in de laatste plaats omdat er een discrepantie lijkt te zijn tussen de manier waarop machine learning algoritmen data het beste aangeboden dienen te krijgen (gebalanceerd) voor een optimale werking en de wijze waarop dit in de praktijk zich presenteert (extreem ongebalanceerd). Het zou mijns inziens mooi zijn wanneer ontwikkelde detectiemethoden sneller hun weg vinden naar de praktijk en daarmee onze maatschappij helpen te verdedigen tegen deze bedreiging. Want dat de gevaren die botnets met zich meebrengen en daarmee onze maatschappij bedreigen, mag duidelijk zijn.

7 CONCLUSIE

In dit hoofdstuk zal naast het beantwoorden van mijn onderzoeksvraag, ook de bijdrage aan het onderzoeksveld worden toegelicht, eventueel toekomstig werk worden besproken en een reflectie worden gegeven op mijn afstudeerproces.

7.1 Beantwoording onderzoeksvraag

Om de hoofdvraag van mijn onderzoek te beantwoorden zullen eerst de subvragen beantwoord worden:

1 Wat zijn botnets en hoe werken ze?

Een botnet is een netwerk van met malware geïnfecteerde computers, bots genaamd, dat op afstand bediend wordt door een of meerdere aanvallers, botmasters, om ingezet te worden voor cybercrime. Elk botnet heeft een infrastructuur waarlangs de commando's worden verstuurd naar de bots en waarmee gecommuniceerd wordt met de botmaster: de Command & Control server. De manier waarop deze communicatie-infrastructuur is georganiseerd onderscheidt de verschillende typen botnets.

Botnets zijn door de jaren heen geëvolueerd van netwerken met vrij eenvoudige, centrale, communicatie structuren tot zeer geavanceerde, hybride, communicatie structuren. Herkenning en ontmanteling is hiermee steeds lastiger geworden. Waar bij een centrale infrastructuur de communicatie direct van botmaster naar bot verloopt, kan een bot bij een hybride structuur zowel de taak van C&C server als bot op zich nemen.
Een bot doorloopt verschillende stadia voordat deze onderdeel uitmaakt van het botnetwerk. Pas nadat een host is geïnfecteerd en zich succesvol heeft aangemeld bij het botnetwerk, is het hiervan onderdeel geworden.

2 Hoe werken de technieken fast-flux en domein flux en hoe worden ze toegepast? Zijn er overeenkomende kenmerken tussen deze twee technieken?

Het basisconcept van fast-flux is de koppeling van een domeinnaam aan meerdere IP-adressen, waarbij een zeer korte Time To Live (TTL) is ingesteld. Elke keer dat een domeinnaam wordt opgevraagd, wordt een ander IP-adres terug gegeven. Wanneer alleen IP-adressen behorend bij een domeinnaam worden veranderd, spreekt men van single-flux. Wanneer ook IP-adressen van een DNS server worden veranderd, wordt dit double-flux genoemd.
Bij domain-flux worden domeinnamen behorend bij een IP-adres in hoog tempo veranderd. Deze domeinnamen worden maar voor korte tijd geregistreerd.

Domain-flux en fast-flux hebben dus geen overeenkomende kenmerken maar zijn tegengesteld aan elkaar. Beide technieken worden door botnets vooral ingezet om ontdekking en ontmanteling te voorkomen. Door snelle wijziging in zowel IP-adressen als domeinnamen zijn Command & Control servers lastig te detecteren en zijn ze ook minder gevoelig voor blacklisting. Tegen de tijd dat een domeinnaam of een IP-adres op een blacklist staat, gebruikt een botmaster alweer een ander IP-adres en/of domeinnaam. Daarnaast wordt bij domein-flux een grote hoeveelheid domeinnamen gegenereerd met behulp van een domain generation algoritme (DGA) waarbij maar een klein deel van deze domeinen voor korte tijd daadwerkelijk wordt geregistreerd. Uit de literatuur komt naar voren dat beide technieken worden toegepast door botnets.

3 Welke taxonomie wordt gebruikt voor botnetdetectie op basis van DNS verkeer en welke machine learning algoritmen worden hierbij vooral toegepast?

Botnetdetectie op basis van DNS verkeer valt onder de categorie passieve netwerk detectiemethoden. Er wordt gedetecteerd op basis van monitoring van netwerkverkeer, in dit geval DNS verkeer. Er wordt niet actief ingegrepen en ook niet in de data pakketten zelf gekeken. Botnetdetectie op basis van DNS verkeer kan worden onderverdeeld in de volgende 5 categorieën:

- Flow: gericht op het classificeren van verkeersstromen.
- Anomaly: gericht op het zoeken naar afwijkingen in verkeerstromen of parameters.
- Flux: gericht analyseren van veranderingen in IP-adressen behorend bij een domeinnaam.
- DGA: gericht op het analyseren van karakteristieken van opgevraagde domeinnamen.
- Botnetinfectie: gericht op detectie van geïnfecteerde hosts.

Binnen botnetdetectie op basis van DNS worden vooral veel machine learning algoritmen gebruikt die geschikt zijn voor clustering of voor classificatie doeleinden.

4 Wat zijn de belangrijkste recursive DNS kenmerken wanneer botnets gebruik maken van fast-flux en domain-flux? Welke features kunnen hieruit worden geselecteerd?

De belangrijkste recursive DNS kenmerken zullen vooral tijdens de connectie/rally fase en de onderhoud/update fase te zien zijn (zie pagina 14: Figuur 3)Figuur 3 DE LEVENSFASEN VAN EEN BOTNET (SILVA ET AL. 2013). In deze fases vindt er communicatie tussen bots en een Command en Control server plaats. Tijdens deze communicatie wordt fast-flux/domain-flux ingezet om herkenning en ontmanteling te voorkomen.

Belangrijkste kenmerken van deze communicatie zijn:

- Het aantal NXdomains-antwoorden, dat wil zeggen antwoorden dat een opgevraagd domein niet bestaat, is significant hoger in een bepaalde tijdsperiode dan wanneer er geen gebruik gemaakt wordt van domain-flux door botnets.
- Domeinnamen gegenereerd met DGA hebben een andere taal-technische opbouw dan legitieme domeinnamen.
- Het aantal DNS requests zal hoger liggen dan in perioden wanneer botnets niet actief zijn.
- Messagelengte van een DNS answer voor een malafide domein is mogelijk anders dan die van een bonafide domeinnaam.

Karakteristieken die na het verrijken van recursive DNS data naar voren komen:

- Geografische spreiding: meer antwoorden waarbij bijbehorende IP-adressen verdeeld zijn over meerdere landen.
- Antwoorden waarbij bijbehorende IP-adressen verdeeld zijn over meerdere ASN nummers.
- Antwoorden waarbij bijbehorende IP-adressen verdeeld zijn over meerdere registrars.
- Antwoorden waarbij domeinnamen minder dan 2 weken bestaan.

De feature TTL kan worden gebruikt als een white-list criteria: domeinen met een hoge TTL kunnen worden beoordeeld als bonafide. Wel moeten worden opgemerkt dat een grote meerderheid van alle domeinen een lage TTL hebben. De aanname dat de feature TTL een goede voorspeller zou zijn van malafide domeinen, is onjuist.

De gekozen features in mijn onderzoek: TTL, msglen, ASN, ASN registrar, Landen waar IP-adressen vandaan komen, aantal verschillende landen, aantal IP-adressen, aantal verschillende ASN nummers, aantal verschillende ASN registrars en de ouderdom van een domein, geven goede resultaten om malafide domeinnamen te herkennen. Uit analyse van de trainingsdata blijkt wel dat niet alle features even relevant zijn. Uit deze analyse blijkt dat de features ouderdom van een domein en msglen het meest relevant zijn. De features aantal verschillende AS nummers en aantal verschillende ASN registrars lijken vooral weinig toe te voegen. Ondanks de goede resultaten is het interessant om nader te onderzoeken of resultaten verbeteren wanneer minder relevante features worden weggelaten.

5 Welke dataset kan gebruikt worden en hoe kan de 'ground truth' hiervan bepaald worden?

Een up-to-date dataset van een ISP netwerk die zowel publiekelijk beschikbaar als gelabeld is, heb ik niet kunnen vinden. Veel datasets zijn verouderd en datasets van ISP netwerken waren nog schaarser en helemaal niet up-to-date. Daarom heb ik besloten 2 datasets op te stellen op basis van eigen data van onze DNS servers. Eén dataset bevat gebalanceerde data met ongeveer gelijke hoeveelheden malafide en bonafide data. De andere dataset is scheef gebalanceerd: de hoeveelheid malafide data is ondervertegenwoordigd. Beide datasets zijn als input gebruikt voor de modellen.

Met het bepalen van de ground truth wordt vastgesteld wat de waarheid is voor het model. Ground truth doet vermoeden dat het een fundamentele waarheid is: voor iedereen duidelijk en onbetwistbaar. In de praktijk betekent het echter dat het een waarheid is die de opsteller van een dataset bepaalt op basis van door hem gebruikte criteria.
Bij het opzetten van een eigen dataset is het dus belangrijk om gebruikte criteria en uitgangspunten goed te beschrijven. Wanneer gebruik gemaakt wordt van reeds bestaande datasets is het van belang deze criteria vooraf duidelijk te hebben zodat een model op de juiste uitgangspunten wordt gebaseerd.

Een veel gebruikte manier om de ground truth te bepalen is het gebruik van blacklists en whitelist. Belangrijk is daarbij te achterhalen op basis van welke criteria een blacklist is opgezet. Immers een blacklist gericht op illegale content is een andere dan één die gericht is op botnetactiviteiten. Aan te bevelen is om meerdere lijsten te gebruiken. Voor mijn onderzoek heb ik uit praktische overwegingen maar één bron gebruikt, namelijk Tesorion. Uit mijn literatuuronderzoek blijkt dat er vrijwel geen overlap tussen blacklists bestaat. Dit betekent dat een blacklist maar een klein deel van het malafide verkeer eruit filtert. Hoewel dit een van de meest gebruikte methoden is, is 100% zekerheid onmogelijk. Dit betekent eigenlijk per definitie dat elke detectiemethode, gebaseerd op een dataset waarvan de ground truth op met behulp van blacklists bepaald is, niet foutloos kan zijn.

Een whitelist is opgebouwd uit de meest populaire domeinnamen. Gedachte hierbij is dat wanneer een domein zeer populair is, het als goedaardig kan worden beschouwd. Whitelists worden gebruikt om de hoeveelheid data die beoordeeld moet worden als malafide, te verkleinen. Voor mijn onderzoek heb ik de whitelist, in mijn geval de Tranco-lijst, niet gebruikt om de hoeveelheid data te verkleinen. Het controleren van alle domeinnamen van 1-12-2020 met de Tranco-lijst en door Tesorion is parallel uitgevoerd. Domeinnamen die door de Tranco-lijst als bonafide zijn aangemerkt en tegelijkertijd door Tesorion als malafide, zijn uiteindelijk als malafide gelabeld.

6 Welk machine learning algoritme kan hierbij gebruikt worden. Wat is de succesratio van dit algoritme en hoe staat dit in verhouding met andere reeds bekende vergelijkbare detectiemethoden?

Voor het conceptuele model heb ik diverse machine learning technieken overwogen en heb ik diverse optimalisatie technieken toegepast. Uiteindelijk is uit mijn experimentele veldonderzoek gebleken dat het Random Forest algoritme de beste resultaten oplevert in vergelijking met de algoritmen k-Nearest Neighbors en Support Vector Machine. Bovendien bleek dat het model het beste presteert wanneer gebruik gemaakt wordt van een gebalanceerde testset (Dataset I). In dit geval is het model in staat om in 94.8% van alle malafide domeinnamen juist te voorspellen (Recall). De algehele effectiviteit is in dit geval 94.3% (Accuracy).
Wanneer gebruik gemaakt wordt van een ongebalanceerde testset (Dataset II) is de Recall score 78.6% en de Accuracy score 62.2%, wat slecht te noemen is. Wanneer dit model opnieuw getest wordt met DatasetI (gebalanceerd), zijn de resultaten beter dan wanneer getraind en getest wordt met DatasetI: Recall score 97.5% en Precision score 97.8%.

Mijn uitkomsten vergelijken met andere DNS detectiemethoden is lastig. De situaties zijn vaak verschillend wat betreft de te detecteren botnets of de data die gebruikt wordt. Ook performance indicatoren worden niet altijd uniform toegepast. Wanneer toch gekeken wordt naar methoden die enigszins vergelijkbaar zijn, dan liggen de uitkomsten in dezelfde lijn.

Met de beantwoording van de deelvragen, kan ook antwoord worden gegeven op de onderzoeksvraag:

***Hoe kunnen botnets die gebruik maken van fast-flux en domain-flux technieken, gedetecteerd worden met behulp van machine learning, waarbij gebruik gemaakt wordt van recursieve DNS data*?**

Uit mijn experimenteel onderzoek blijkt dat het mogelijk is om op basis van onze eigen recursive DNS data botnets te detecteren die gebruik maken van fast-flux en domain-flux. Het detecteren kon met behulp van een tiental features: TTL, MSGlen (lengte van het bericht), ASN, ASN registrar, land van afkomst van een IP-adres, aantal IP-adressen, aantal ASN nummers, aantal ASN registrars, aantal landen van afkomst van IP-adressen en ouderdom van een domeinnaam. Niet alle features zijn hierbij even relevant en nader onderzoek hiernaar is gewenst.
De beste resultaten werden verkregen met een model gebouwd op het Random Forest algoritme en een gebalanceerde dataset. Met een niet gebalanceerde dataset verslechteren de resultaten. Hoe mijn model in de praktijk gaat werken met zijn zeer scheef gebalanceerde verhoudingen tussen malafide en bonafide domeinnamen, is dan ook niet met zekerheid te zeggen.

7.2 Bijdrage van mijn onderzoek

Het aantal features in mijn onderzoek is klein te noemen in vergelijking tot andere botnetdetectiemethoden. Wel blijkt dat met slechts tien features goede resultaten geboekt kunnen worden. Mijn onderzoek heeft geleid tot een nieuwe set van features die, bij mijn weten, nog niet eerder is gebruikt.

Mijn onderzoek bevestigt dat machine learning modellen niet goed werken met ongebalanceerd data. Echter, in de botnetdetectiepraktijk van alle dag is de realiteit dat data zeer scheef is gebalanceerd.
Mijn onderzoek heeft ook inzichtelijk gemaakt dat niet duidelijk is hoe verschillende detectiemethoden in de praktijk presteren.

Mijn onderzoek geeft een blauwdruk hoe een dataset samengesteld kan worden waarmee malafide domeinnamen kunnen worden gedetecteerd.

7.3 Discussie

De resultaten van mijn onderzoek laten zien dat fast-flux en domain-flux botnets gedetecteerd kunnen worden met een relatief klein aantal features en dat recursive DNS data hier zeker geschikt voor is. Er zit hier wel een aantal 'maren' aan vast:

- Om te beginnen, het beschikbaar zijn van datasets. Voor mijn onderzoek was er geen passende dataset aanwezig. Het maken van de eigen dataset was een zeer tijdrovende bezigheid. Validatie van malafide domeinnamen door meerdere partijen heeft de voorkeur, om praktische redenen heb ik mij beperkt tot één partij.
- Het is goed te beseffen dat het bepalen van de ground truth op basis van blacklisting en whitelisting nooit helemaal waterdicht is. Omdat gebruik gemaakt wordt van input van derden waarbij niet inzichtelijk is hoe de labeling tot stand is gekomen. Er van uitgaand dat deze blacklisting en whitelisting ook worden opgesteld op basis van algoritmen, is het mogelijk dat hierin een bias zit. Deze zal dan doorwerken in een eigen dataset.
  Ook kunnen in blacklisten domeinen opgenomen zijn die wel illegale activiteiten bevatten maar mogelijk geen botnetactiviteiten zijn. Ground truth bepaling van een dataset bij botnetdetectie bevat dus altijd een zekere mate van subjectiviteit.
- De periode die mijn datasets omvatten is kort: één dag. Niet alle bekende fast-flux en domain-flux botnets zullen in mijn datasets aanwezig zijn. Het onderzoek geeft een indicatie dat de door mij gekozen opzet goed is.

Wat mij heeft geïntrigeerd is de tegenstelling die er is tussen hoe machine learning algoritmen het beste data kunnen verwerken, namelijk gebalanceerd, versus hoe data daadwerkelijk in de praktijk voorkomt: ongebalanceerd. Een algoritme kan slecht overweg met data die scheef verdeeld is. In de praktijk zijn malafide en bonafide verkeer absoluut niet evenredig verdeeld. Uit mijn literatuuronderzoek werd duidelijk dat dit probleem in een aantal onderzoeken wel zijdelings wordt genoemd. Krawczyk en Viloria et al. beschrijven dit uitvoeriger en constateren dat huidige technieken om een dataset te balanceren niet berekend zijn op de scheve verhoudingen binnen het gebied van botnetdetecite (Krawczyk, 2016), (Viloria, Lezama, & Mercado-Caruzo, 2020).
Blijft bij mij de vraag hoe goed dan deze huidige detectie oplossingen in de praktijk werken.

7.4 Suggesties voor nader onderzoek

Voor de toekomst zijn er diverse onderwerpen die interessant zijn voor nader onderzoek:

- *Het nader onderzoeken van gebruikte features:* om praktische redenen heb ik niet onderzocht of alle gebruikte features nodig waren. Wanneer bepaalde features niet voldoende waarde toevoegen aan het model, kunnen ze ruis veroorzaken. Resultaten van het model kunnen verbeterd worden door deze dan te verwijderen. Het kan ook zinvol zijn te onderzoeken wat de resultaten worden als andere technieken voor normalisatie gebruikt worden.
- *Het omgaan met bigdata*: bij machine learning geldt vaak dat hoe meer data hoe beter een model kan voorspellen. Het werken met big data vereist veel computerkracht die niet altijd voorhanden is. Voor mijn onderzoek was meer dan genoeg data beschikbaar. Echter, het transformeren van deze hoeveelheden data naar een bruikbare dataset eiste dermate veel computerkracht dat dit niet haalbaar was. Daardoor zijn er concessies gedaan in de omvang van mijn datasets. Het is interessant te onderzoeken of er manieren zijn om DNS logfiles te verwerken op andere wijzen die minder computerkracht kosten.
- *Publiekelijk beschikbare datasets*. Tijdens mijn literatuuronderzoek heb ik meerdere datasets onderzocht, weinigen zijn recent. Aanwezige datasets zijn veelal niet verrijkt met aanvullende informatie zoals GEO informatie, bijbehorende ASN nummers of ouderdom van een domeinnaam. Deze informatie kan lastig achteraf worden verkregen wanneer data te zeer verouderd is. Voor het onderzoeksgebied botnetdetectie zou het

realiseren van nieuwe zo compleet mogelijke datasets een goede aanvulling zijn. Daarnaast zou het ook wenselijk zijn een dataset te creëren op een productie omgeving. Hierbij is het belangrijk ook het juridisch kader te onderzoeken. Met de intrede van nieuwe privacy richtlijnen zijn regels omtrent privacy strikter geworden. Een IP-adres wordt in bepaalde gevallen gezien als een persoonsgegeven. Duidelijkheid omtrent wat juridisch wel en niet mag, is belangrijk voor het opzetten van nieuwe datasets.

- *Onderzoek versus praktijk*: Er is veel literatuur onderzoek gedaan naar diverse manieren van botnetdetectie en de werking van een model. Onderzoek naar botnetdetectie in de praktijk met real-life opstellingen ben ik weinig tegen gekomen. Het is interessant om diverse ontworpen modellen te valideren in dagelijkse situaties. Een belangrijk onderdeel hiervan is de extreem scheve verhouding tussen malafide en bonafide domeinen danwel verkeer. Machine learning algoritmen zijn niet geschikt voor ongebalanceerde datasets en methoden om deze gebalanceerd te krijgen, gaan uit van verhoudingen van 1:4 tot 1:100. De verhouding in de praktijk is vaker tussen de 1:1000 en 1:5000. Naar deze praktijksituatie van extreem ongebalanceerde datasets is nog onvoldoende onderzoek gedaan.
- *Impact van DNS over Https (DoH) / DNS over TLS (DoT) op botnetdetectie*: DoH en DoT zijn technieken om communicatie met een DNS server te beveiligen. Botnets kunnen deze technieken ook gebruiken om hun communicatie met een DNS server te versleutelen. Dit heeft een grote impact op het detecteren van botnets op basis van DNS. Volgens ander andere Hjelm (Hjelm, 2019) en Patsakis et al. (Patsakis, Casino, & Katos, 2020) zullen deze technieken steeds vaker door botnets gebruikt worden. Het is zeer gewenst dat onderzoek naar deze technieken en de gevolgen hiervan voor botnetdetectie gaat plaatsvinden.

7.5 Reflectie

Na tien jaar toch het afstuderen weer oppakken, was pittig. De tijd heeft niet stil gestaan en de onderwerpen die ik voor mijn scriptie gekozen had, heb ik mij helemaal eigen moeten maken. Wel zocht ik, anders dan toen ik alleen student was, veel sneller contact binnen en buiten mijn netwerk. Kennis kwam niet alleen meer uit (digitale) bronnen, maar ook vond ik mensen die mij weer verder op weg konden helpen. Tijdens mijn afstudeerperiode heb ik kennis gemaakt met veel verschillende mensen uit diverse disciplines: politie, onderzoekers, data scientists, etc. Zowel binnen Nederland als daarbuiten. Deze contacten hebben mij verrijkt en mijn netwerk verder versterkt. Naast de toename van kennis zijn deze contacten en momenten voor mij net zo waardevol geweest. Ik kijk hier met veel plezier op terug.

Het idee om met een eigen dataset te gaan werken, heeft mij absoluut veel kopzorgen gegeven en zo nu en dan ook tot wanhoop gedreven. Hoe bepaal je de ground truth eigenlijk? En wanneer je daar een idee van hebt, blijkt dat het verkrijgen van blacklisten lastiger is dan vooraf gedacht. Het verrijken van de data kostte dan ook heel veel doorlooptijd. Hoe langer het duurde, hoe ouder mijn datasets werden. Hiermee liep ik het gevaar dat bepaalde verrijkingsslagen niet meer mogelijk waren. Meerdere malen heb ik moeten besluiten opnieuw data op te slaan omdat de oude niet meer bruikbaar was om te verrijken. Dit onderdeel heeft de doorlooptijd van mijn afstudeertraject behoorlijk verlengd. Vraagstukken waar ik voor stond waren gelukkig niet uniek en werden ook herkend door onder andere mijn afstudeerbegeleider. Dankzij zijn feedback en het overleggen over diverse opties gaf hij mij het vertrouwen om door te gaan.

De richting van afstuderen: “iets met botnets en iets met machine learning”, had ik snel gevonden. In het begin vond ik het lastig een onderzoeksvraag te formuleren. Ik had het idee dat bijna alles al wel een keer onderzocht was. Nu, na afloop van het hele traject, ben ik me bewust van de vele onderwerpen die nog niet onderzocht zijn. En dat mijn onderzoek een klein

steentje aan dit onderzoeksveld mag bijdragen. Wel heb ik mij behoorlijk verkeken op de hoeveelheid werk. Want niet alleen moest ik kennis opdoen over de werking van DNS en over botnets en botnetdetectie maar ook machine learning was geheel nieuw voor mij. Tijdens het vooronderzoek werd duidelijk dat mijn kennis op dit gebied ernstig te kort schoot. Na het volgen van een aantal cursussen, het doen van literatuuronderzoek en het benaderen van mensen die mij in korte tijd kennis hebben bijgebracht, heb ik voldoende bagage gekregen om het onderzoek af te ronden.

Het onderzoek heeft mij ook een kritische blik opgeleverd ten op zichtte van machine learning in het algemeen en het gebruik van machine learning bij botnetdetectie in het bijzonder. Wij als ontwikkelaars van dergelijke systemen hebben een grote verantwoordelijkheid. Machine learning als vakgebied biedt veel mogelijkheden, maar om aan sluiten bij de woorden van Ada Lovelace, dergelijke systemen kunnen niet zelf denken. Mogelijkheden zijn groot maar niet eindeloos en wij mensen sturen deze systemen meer dan dat we ons bewust zijn. Met de data die we kiezen, de manier waarop we labelen, de keuze van algoritmen en leermodellen, bepalen wij de richting in plaats van het systeem.

Het schrijven van deze scriptie heeft de nodige zweetdruppels gekost. Ik heb gemerkt dat het eerst nadenken over het raamwerk hier ontzettend bij helpt. Het geeft richting in de uitvoering van de werkzaamheden. Daarnaast hebben de vele gesprekken met mijn buurman dhr. D.F. van Giffen mij geleerd om heldere teksten te schrijven die ook voor een geïnteresseerde, en niet inhoudelijk onderlegde, lezer toegankelijk zijn. Kritisch te zijn op wat je wel wilt zeggen en wat overbodig is.

Het is een lange zoektocht geweest. De combinatie van gezin, eigen bedrijven en studie was pittig en niet altijd eenvoudig. Uiteindelijk ben ik echt op gang gekomen nadat ik focus had aangebracht, keuzes had gemaakt waar ik tijd aan kon besteden. Waar ik altijd geneigd ben om te denken “het kan allemaal wel”, kon dat nu niet. Er is een periode geweest dat ik gedacht heb dat afronding er niet in zat. Maar nu hier deze reflectie schrijvend, constateer ik dat het een enorm leerzaam traject was, waarbij ik leerde kritisch om mij heen te kijken, mij te verwonderen wat er allemaal kan en nieuwe kennis en kunde heb aangeleerd.

ACADEMISCHE REFERENTIES

Acarali, D., Rajarajan, M., Komninos, N., & Herwono, I. (2016). Survey of approaches and features for the identification of HTTP-based botnet traffic. *Journal of Network and Computer Applications, 2016*(76), 1-15. doi:10.1016/j.jnca.2016.10.007

Alieyan, K., Almomani, A., Manasrah, A., & Kadhum, M. (2017). A survey of botnet detection based on DNS. *Neural Comput & Applications, 28*(7), 1541-1558. doi:10.1007/s00521-015-2128-0

Almomani, A. (2018). Fast-flux hunter: a system for filtering online fast-flux botnet. *Neural Computing and Applications, 29*(7), 483-493. doi:10.1007/s00521-016-2531-1

Amin, A., Anwar, S., Adnan, A., Nawaz, M., Howard, N., Qadir, J., . . . Hussain, A. (2016). Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study. *IEEE Access*, 7940-7957. doi:10.1109/ACCESS.2016.2619719

Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., & Feamster, N. (2010). Building a Dynamic Reputation System for DNS. *Proceedings of the 19th USENIX conference on Security (SEC'10) 11-13 August 2010* (pp. 273-290). Washington DC,: Usenix Associaton. doi:10.5555/1929820.1929844

Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou, N., & Dagon, D. (2011). Detecting Malware Domains at the Upper DNS Hierarchie. *Proceedings of the 20th USENIX conference on Security (SEC'11) 08-12 August 2011* (pp. 27-27). Berkeley, CA, USA: USENIX Association. doi:10.5555/2028067.2028094

Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., & Dagon, D. (2012). From throw-away traffic to bots: Detecting the rise of DGA-based malware. *Proceedings of the USENIX Conference on Security Symposium* (p. p.24). Bellevue, USA: USENIX. doi:10.5555/2362793.2362817

Asghari, H., Van Eeten, M. J., & Bauer, J. M. (2015). Economics of Fighting Botnets: Lessons from a Decade of Mitigation. *IEEE Security & Privacy, 13*(5), 16-23. doi:10.1109/MSP.2015.110

Bilge, L., Kirda, E., Kruegel, C., & Balduzzi, M. (2011). EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. *Proceedings of the Network and Distributed System Security Symposium*. San Diegeo, California: NDSS.

Bilge, L., Sen, S., Balzarotti, D., Kirda, E., & Kruegel, C. (2014). EXPOSURE: A Passive DNS Analysis Service to Detect and Report Malicious Domains. *ACM Transactions on Information and System Security, 16*(4), 28. doi:10.1145/2584679

Caglayan, A., Toothaker, M., Drapeau, D., Burke, D., & Eaton, G. (2009). Real-Time Detection of Fast Flux Service Networks. *Cybersecurity Applications & Technology Conference For Homeland Security, Catch'09* (pp. 285-292). Washington, DC, USA: IEEE. doi:10.1109/CATCH.2009.44

Choi, H., & Lee, H. (2012). Identifying botnets by capturing group acitvities in DNS traffic. *Computer Networks, 56*(1), 20-33. doi:10.1016/j.comnet.2011.07.018

Damasevicius, R., Venckauskas, A., Grigaliunas, S., Toldinas, J., Morkevicius, N., Aleliunas, T., & Smuikys, P. (2020). LITNET-2020: An Annotated Reals-World Network Flow Dataset for Network Intrusion Detection. *Electronics*, 800. doi:10.3390/electronics9050800

Doorewaard, H., & Verschuren, P. (2015). *Het ontwerpen van een onderzoek* (vijfde druk ed.). Amsterdam: Boom Lemma Uitgevers, ISBN: 9789462365070.

Garcia, S., Zunino, A., & Campo, M. (2014). Survey on network-bases botnet detection methods. *Security and communication networks, 2014*(7), 878-903. doi:10.1002/sec.800

Grzinic, T., Perhoc, D., Maric, M., Vlasic, F., & Kulcsar, T. (2014). CROFlux - Passive DNS method for detecting fast-flux domains. *37th International convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 26-30 May 2014* (pp. 1376-1380). Opatija, Croatia: IEEE. doi:10.1109/MIPRO.2014.6859782

Hands, N., Yang, B., & Hansen, R. (2015). A study on botnets utilizing DNS. *Proceedings of the 4th annuyal ACM conference on research in information technology* (pp. 23-28). New York: Association for Computing Machinery. doi:10.1145/2808062.2808070.

Hoang, X., & Nguyen, Q. (2018). Botnet Detection Based on Machine Learning Techniques Using DNS Query Data. *Future Internet, 10*(5), 43. doi:10.3390/fi10050043

Hossin, M., & Sulaiman, M. (2015). A Review on Evaluation Metrix For Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process, 5*(2), 1-11. doi:10.5121/ijdkp.2015.5201

Hu, X., Knysz, M., & Shin, K. (2011). Measurement and Analysis of Global UP-Usage Patterns of Fast-Flux Botnets. *Proceedings IEEE INFOCOM, 10-15 April 2011* (pp. 2633-2641). Shanghai, China: IEEE. doi:10.1109/INFCOM.2011.5935091

Hyslip, T., & Pittman, J. (2015). A Survey of Botnet Detection Techniques by Command and Control Infrastructure. *Journal of Digital Forencsics, Security and Law, 10*(1), Article 2. doi:https://doi.org/10.15394/jdfsl.2015.1195

Jazi, H., Gonzalez, H., Stakhanova, N., & Ghorbani, A. (2017). Detecting HTTP-based Application Layer DoS attacks on Web Servers in the presence of sampling. *Computer Networks, 121*, 25-36. doi:10.1016/j.comnet.2017.03.018

Jhaveri, M. H., Cetin, O., Gañán, C., & Van Eeten, M. (2017). Abuse Reporting and the Fight Against Cybercrime. *ACM Computer Surveys, 49*(4), 1-27. doi:10.1145/3003147

Jordan, M., & Mitchell, T. (2015, juli 17). Machine Learning: Trends, perspectives, and prospects. *Schience*, 255-260. doi:10.1126/science.aaa8415

Khattak, S., Ahmed, Z., Syed, A., & Khayam, S. (2015). BotFlex: A community-driven tool for botnet detection. *Journal of Netwerk and Computer Applications, 2015*(58), 144-154. doi:10.1016/j.jnca.2015.10.002

Kheir, N., Tran, F., Caron, P., & Deschamps, N. (2014). Mentor: Positive DNS Reputation to Skim-off Benign Domains in Botnet CC Blacklists. *29th IFIP INternational INformation Security Conference (SEC), june 2014* (pp. 1-14). Marrackec, Marrokko: Springer. doi:10.1007/978-3-642-55415-5_1

Krawczyk, B. (2016). Learning form imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence, 5*, 221-232. doi:10.1007/s13748-016-0094-0

Le Pochat, V., Van Goethem, T., Tajalizadehkoob, S., Korczynski, M., & Joosen, W. (2019). Tranco: A Research-Oriented Top Sites Ranking Hardenend Against Manipulation. *Proceedings of the 26th Annual Netwok and Distributed System Security Symposium (NDSS 2019)* (pp. -). San Diego: -. doi:10.14722/ndss.2019.23386

Li, X., Wang, J., & Zhang, X. (2017). Botnet Detection Technology Based on DNS. *Future internet, 9*(4), 55. doi:10.3390/fi9040055

Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., & Therón, R. (2018). UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers & Security, 73*, 411-424. doi:10.1016/j.cose.2017.11.004

Mahmoed, M., Nir, M. P., & Matrawy, A. (2014). A Survey on Botnet Architectures, Detection and Defences. *International Journal of Network Security, 17*, -.

Mierswa, I., Klinkenberg, R., Fischer, S., & Ritthof, O. (sd). A Flexible Platform for Knowledge Discovery Experiments: YALE - Yet Another Learning Environment. Opgehaald van http://www-ai.cs.uni-dortmund.de:8080/PublicPublicationFiles/mierswa_etal_2003a.pdf

Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., García, Á., Heredia, I., . . . Hluchý, L. (2019). Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Rev*(52), 77-124. doi:https://doi.org/10.1007/s10462-018-09679-z

Passerini, E., Paleari, R., Martignoni, L., & Bruschi, D. (2008). FluXOR: Detecting and Monitoring Fast-Flux Service Networks. *Zamboni D. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2008. Lecture Notes in Computer Science. 5137*, pp. 186-206. Berlin, Heidelberg: Springer. doi:10.1007/978-3-540-70542-0_10

Patsakis, C., Casino, F., & Katos, V. (2020). Encrypted and cover DNS queries for botnets: Challenges and coutermeasures. *Computers & Security, 88*. doi:10.1016/j.cose.2019.101614

Pedregosa, F., Varquaux, G., Gramfort, A., Mlchel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Perdisci, R., Corona, I., Dagon, D., & Lee, W. (2009). Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces. *Computer Security Applications Conference, Annual (ACSAC)* (pp. 311-320). Honolulu, Hawaii: IEEE. doi:10.1109/ACSAC.2009.36

Ring, M., Wunderlich, S., Gruedl, D., Landes, D., & Hotho, A. (2017, 12 22). Creation of Flow-Based Data Sets for Intrusion Detection. *Journal of Information Warfare*, 40-53. Opgehaald van https://www.researchgate.net/publication/322337341_Creation_of_Flow-Based_Data_Sets_for_Intrusion_Detection

Ring, M., Wunderlich, S., Gruedl, D., Landes, D., & Hotho, A. (2017). Flow-based benchmark data sets for intrusion detection. *Proceedings of the 16th European Conference on Cyber Warfase and Security (ECCWS)* (pp. 362-369). Dublin: ACPIL.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computer & security, 2019*(86), 147-167. doi:10.1016/j.cose.2019.06.005

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review, 65*(6), 386-408. doi:10.1037/h0042519

Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development, 3*(3), 210-229. doi:10.1147/rd.33.0210

Schiavoni, S., Maggi, F., Cavallaro, L., & Zanero, S. (2014). Phoenix: DGA-Based Botnet Tracking and Intelligence. *Dietrich S. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2014* (pp. 417-423). Cham, Switzerland: Springer. doi:10.1007/978-3-319-08509-8_11

Sharafaldin, I., Lashkari, A., & Ghorbani, A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *Proceedings of the international conference on information systems security and privay (ICISSP)* (pp. 108-16). Portugal: SciTePress. doi:10.5220/0006639801080116

Sharifnya, R., & Abadi, M. (2015). DFBotKiller: Domain-flux botnet detection based on the history of group activities and failures in DNS traffic. *Digital Investigation, 2015*(12), 15-26. doi:10.1016/j.diin.2014.11.001

Sharma, R., Singla, R., & Guleria, A. (2018). A new labeled flow-based DNS dataset for anomaly detection: PUF dataset. *Procedia Computer Science, 132*, 1458-1466. doi:10.1016/j.procs.2018.05.079

Silva, S., Silva, R., Pinto, R., & Salles, R. (2012). Botnets: A survey. *Computer Networks, 57*(2), 378-403. doi:ISSN: 1389-1286, DOI: 10.1016/j.comnet.2012.07.021

Singh, M., Singh, M., & Kaur, S. (2019). Issues and challenges in DNS based botnet detection: A survey. *Computers & Security, 2019*(86), 28-52. doi:10.1016/j.cose.2019.05.019

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management, 45*, 427-437. doi:10.1016/j.ipm.2009.03.002

Stevanovic, M., & Pedersen, J. (2013). *Machine Learning for identifying botnet network traffic*. Faculty of Engineering and Science. Aalborg East: Aalborg University.

Stevanovic, M., Pedersen, J., D'Alconzo, A., Ruehrup, S., & Berger, A. (2015). On the ground truth problem of malicious DNS traffic analysis. *Computers & security, 55*, 142-158. doi:10.1016/j.cose.2015.09.004

Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., . . . Vigna, G. (9-13 november 2009). Your Botnet is My Botnet: Analysis of a Botnet Takeover. *CCS'09 Proceedings of the 16th ACM conference on Computer and Communication Security* (pp. 635-647). Chicago: ACM. doi:10.1145/1653662.1653738

Truong, D.-T., & Cheng, G. (2016). Detecting domain-flux botnet based on DNS traffic features in managed network. *Security and Communication Networks, 9*, 2338-2347. doi:10.1002/sec.1495

Turing, A. (1950). Computing Machinery and Intelligence. *Mind, LIX*(236), 433-460. doi:10.1093/mind/LIX.236.433

van Eeten, M., Bauer, J., Asghari, H., Tabatabaie, S., & Rand, D. (2010). *The role of Internet Service Providers in Botnet Mitigation an Empirical Analysis Based on Spam Data*. Paris: OECD Publishing. doi:https://doi.org/https://doi.org/10.1787/5km4k7m9n3vj-en

Van Roosmalen, J. (2017). *The feasibility of deep learning approaches for P2P-botnet detection*. Faculty of Management, Science and Technology. -: Open Universiteit Nederland.

Viegas, E., Santin, A., & Oliveira, L. (2017). Towards a reliable anomaly-based intrusion detection in real-world environments. *Computer Networks*, 200-216. doi:https://doi.org/10.1016/j.comnet.2017.08.013

Viloria, A., Lezama, O., & Mercado-Caruzo, N. (2020). Unbalanced data processing using oversampling: Machine learning. *The 17th International Conference on Mobile Systems and Pervasive Computing, August 9-12* (pp. 108-113). Belgium: Elsevier. doi:https://doi.org/10.1016/j.procs.2020.07.018

Wang, T.-S., Lin, H.-T., Cheng, W.-T., & Chen, C.-Y. (2017). DBOD: Clustering and detecting DGA-based botnets using DNS traffic analysis. *Computers & Security, 64*, 1-15. doi:10.1016/j.cose.2016.10.001

Wang, Z., Liu, K., Li, J., Zhu, Y., & Zhang, Y. (2019). Various Frameworks and Libraries of Machine Learning and Deep Learning: A Survey. *Archives of Computational Methods in Engineering*, 1886-1784. doi:10.1007/s11831-018-09312-w

Wilkinson, M. (2016). The FAIR Guiding PRinciples for scientific dta management and stewardship. *Scientific Data*, -. doi:10.1038/sdata.2016.18

Xu, C., Shen, J., & Du, X. (2019). Detection method fo domain names genereaded by DGAs based on semantic representation ad deep neural network. *Computers & security, 85*, 77-88. doi:10.1016/j.cose.2019.04.015

Zhauniarovich, Y., Khalil, I., Yu, T., & Dacier, M. (2018). A Survey on Malicious Domains Detection through DNS Data analysis. *ACM Computing Surveys, 51*(4), 1-36. doi:10.1145/3191329

NIET ACADEMISCHE REFERENTIES

Akamai. (2017). *Digging Deeper - an In-Depth Analysis of a Fast Flux Network*. Las Vegas, Nevada: Akamai. Opgeroepen op 08 10, 2018, van https://www.akamai.com/us/en/about/news/press/2017-press/fast-flux-botnets-still-wreaking-havoc-on-internet-according-to-akamai-research.jsp en https://www.akamai.com/us/en/multimedia/documents/white-paper/digging-deeper-in-depth-analysis-of-fast-flux-net

Alexa. (2020). *Alexa Top Sites*. Opgeroepen op 12 14, 2020, van Alexa: https://www.alexa.com/topsites

CBS. (2019). *Cybersecuritymonitor 2019*. Den Haag/Heerlen/Bonaire: Centraal Bureau voor de Statistiek. Opgehaald van https://www.cbs.nl/nl-nl/publicatie/2019/37/cybersecuritymonitor-2019

CBS. (2021, 01 13). *ICT-gebruik bij bedrijven; bedrijfstak en bedrijfsgrootte, 2020*. Opgeroepen op 01 28, 2021, van CBS: https://www.cbs.nl/nl-nl/cijfers/detail/84831NED?q=internet#PersoneelDatWerktMetInternet_2

ENISA. (2020). *ENISA Threat Landscape 2020 - Botnet*. Attiki: ENISA. doi:10.2824/552242

Geolocation DB. (2020, 12 20). *Geolocation DB*. Opgehaald van Geolocation DB: https://geolocation-db.com/

Géron, A. (2017). *Hands-on Machine Learning with Scikit-Learn and TensorFlow*. Sebastopol, USA: O'Reilly Media.

Hane, P. (2020, 09 17). *IPWhoIs*. Opgeroepen op 01 10, 2021, van Pypi: https://pypi.org/project/ipwhois/

Hjelm, D. (2019). *A New Needle and Haystack: Detecting DNS over HTTPS Usage*. -: Sans institute. Opgehaald van https://www.sans.org/reading-room/whitepapers/dns/needle-haystack-detecting-dns-https-usage-39160

IETF. (1987, 11 -). *Domain names - Concepts and facilities*. Opgeroepen op 01 22, 2021, van https://www.ietf.org/rfc/rfc1034.txt

IETF. (1987, 11 -). *Domain names - implementation and specification*. Opgeroepen op 01 22, 2021, van https://www.ietf.org/rfc/rfc1035.txt

Koops, P. D.-J. (2013). Acties tegen botnets door SURFnet en bij SURFnet aangesloten instellingen: strafrechtelijke aspecten. April. Opgeroepen op 02 22, 2018, van https://www.surf.nl/binaries/content/assets/surf/nl/kennisbank/2013/expert_opinion_botnets_koops_mei_2013.pdf

Leenes, P. D. (2013, oktober). Acties tegen botnets door SURFnet en bij SRFnet aangesloten instellingen: privacy & data protectie aspecten. Opgeroepen op 02 22, 2018, van https://www.surf.nl/binaries/content/assets/surf/nl/kennisbank/2013/expert_opinion_botnets_leenes_oktober_2013.pdf

Mindmatters. (2020, 05 19). *Lovelace: The programmer who spooked Alan Turing*. Opgeroepen op 03 22, 2021, van Mindmatters: https://mindmatters.ai/2020/05/lovelace-the-programmer-who-spooked-alan-turing/

Mitchel, T. (1997). *Machine Learning*. McGraw-Hill Science. Opgehaald van https://www.cs.ubbcluj.ro/~gabis/ml/ml-books/McGrawHill%20-%20Machine%20Learning%20-Tom%20Mitchell.pdf

Modderkolk, H. (2017, 11 10). *de Volkskrant*. Opgeroepen op 11 12, 2019, van https://www.volkskrant.nl/tech/de-belofte-van-sociale-media-was-dat-ze-ons-blikveld-zouden-verruimen-maar-het-tegenovergestelde-gebeurde~a4535914/

NBIP. (2020, 11 -). DDoS aanvallen in het derde kwartaal van 2020. Opgeroepen op 01 04, 2021, van NBIP - Kenniscentrum: https://www.nbip.nl/wp-content/uploads/2020/11/NBIP-Infographic-DDoS-data-2020-Q3.pdf

NCTV. (2020). *Cybersecuritybeeld Nederland 2020*. Den Haag: Nationaal Coördinator Terrosrismebestrijding en Veiligheid. Opgehaald van https://www.ncsc.nl/documenten/publicaties/2020/juni/29/csbn-2020

Ng, A. (-). Machine Learning. Opgeroepen op 2020, van https://www.coursera.org/learn/machine-learning

Nieuwsuur. (2017, 11 7). *NIeuwsuur*. Opgeroepen op 06 15, 2020, van https://nos.nl/nieuwsuur/artikel/2201690-trollen-en-bots-beinvloeden-ook-jouw-mening-zonder-dat-je-het-merkt.html

NOS. (2020, 02 05). *Hackers Universiteit Maastricht zaten maanden in netwerk: 200.000 euro betaald*. Opgeroepen op 12 23, 2020, van NOS Nieuws: https://nos.nl/artikel/2321732-hackers-universiteit-maastricht-zaten-maanden-in-netwerk-200-000-euro-betaald.html

Pointer, R. (2018, - -). *Eggheads.org*. Opgeroepen op 10 20, 2020, van http://www.eggheads.org

Scikit-learn. (2011, - -). *scikit learn*. Opgeroepen op 02 01, 2021, van Scikit-learn: machine learning in python: www.scikit-learn.org

The Ant Lab. (2019, 05 23). *Dnsanon: extract DNS traffic from pcap to text with optionally anonymization*. Opgeroepen op 11 01, 2020, van The ANT Lab: Analysis of Network Traffic: https://ant.isi.edu/software/dnsanon/

The Honeynet Project. (2008, 08 16). *Know Your Enemey: Fast-Flux Service Networks*. Opgeroepen op 03 24, 2020, van http://www.honeynet.org/papers/ff/

The Spamhaus project. (1998-2021, - -). Opgeroepen op 01 20, 2021, van https://www.spamhaus.org/statistics/botnet-cc/

Tranco. (2021, - -). Opgeroepen op 01 04, 2021, van Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation: https://tranco-list.eu/

Wikipedia. (2021, 02 11). *Ada Lovelace*. Opgeroepen op 04 19, 2021, van Wikipedia: https://nl.wikipedia.org/wiki/Ada_Lovelace